REPRESENTATION LEARNING ON SEQUENTIAL MEDICAL DATA

A Dissertation

Presented to the Faculty of the Weill Cornell Graduate School

of Medical Sciences

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Stephanie L. Hyland

May 2019

REPRESENTATION LEARNING ON SEQUENTIAL MEDICAL DATA

Stephanie L. Hyland, Ph.D.

Cornell University 2019

The way we do medicine is undergoing a revolution driven by technology. As the modern drive to record, share, and analyse data sweeps across society, healthcare lies squarely in its path. Data generated by every-day clinical practice presents an invaluable view of health and disease at a scale previously unimaginable. However, to benefit from it, we need computational tools to extract meaning, clinical insight, and actionable predictions. This new digital era of medicine is an opportunity not only for healthcare providers, but also for machine learning researchers to develop new methods tailored to the unique demands of this complex domain. The work described here sits in this sphere.

Firstly, we explore representation learning for medical language. With its long-tailed distribution of technical terms, medical language necessitates development of methods to augment data-scarcity by exploiting prior information encoded in knowledge graphs. Obtaining semantically meaningful representations of medical concepts and their relationships is vital, and we describe a probabilistic model to learn such representations.

Secondly, we address learning from long time series using recurrent neural networks. These long sequences are commonplace in medicine, where one's health history is necessarily lengthy, but early events nonetheless provide crucial context. To address vanishing and exploding gradients in the training these networks, we propose a novel parametrisation exploiting the correspondence between the Lie group of unitary matrices and its Lie algebra.

Next, a method for generating synthetic ICU time series data is described in the framework of adversarial networks. A core challenge for researchers in healthcare is the scarcity of shareable datasets on which to benchmark. Realistic synthetic data is therefore key. Novel methods for evaluating the quality of this synthetic data are proposed, and the model's privacy and memorisation properties are analysed, both heuristically and in terms of differential privacy.

Finally, an ensemble of gradient-boosted decision trees are employed to identify circulatory system deterioration in Swiss ICU patients. As this system has been developed for deployment, we carefully detail the data processing steps, task specification, and evaluation considerations necessary for a real-world, real-time early warning system driven by machine learning.

BIOGRAPHICAL SKETCH

Stephanie L. Hyland is from Dublin, Ireland. She studied Theoretical Physics at Trinity College Dublin between 2008 and 2012, where she was awarded a BA(Mod.). She then undertook Part III of the Mathematical Tripos at Cambridge University, where she studied Theoretical Physics and Applied Mathematics. During this time, while supported by the Bill and Melinda Gates Foundation, she started to seek research directions outside of theoretical physics, looking for a way to have more immediate impact on society. The conclusion of this deliberation was the decision to travel to the USA in 2013 to join the Tri-Institutional Training Program in Computational Biology and Medicine at Cornell University, Weill Cornell Medical, and Memorial Sloan Kettering Cancer Center. Here, she learned about how machine learning could be used for problems in fundamental biology and medicine, and how physics and mathematics could be used for problems in machine learning. In 2014 she moved from Cornell's home town of Ithaca to its medical campus in New York City, where she joined the group of Gunnar Rätsch, who would become her supervisor. In 2016 the Rätschlab moved to ETH Zürich, and so she returned to her home of Europe to complete her PhD studies, retaining her affiliation with the CBM program.

for Mary Walsh and Keith Hyland

ACKNOWLEDGEMENTS

This would have not been possible without the support of my adviser Gunnar Rätsch, who afforded me the freedom to develop as a scientist, and who believed in me when I didn't believe in myself.

I would like to express my gratitude to my examination committee for their guidance, wisdom, and patience, especially reading the next several hundred pages. They are Adam Siepel, Olivier Elemento, and Thomas Fuchs.

Throughout these years I have had colleagues in New York and Zürich, and who collectively create the welcoming and open atmosphere of the Rätschlab. Through our discussions over lunch, in lab meetings, at conferences and retreats, I have learned from you all and grown both as a scientist and as a person.

In particular, I would like to thank:
Theofanis Karaletsos for many discussions on life, the universe, and modelling,
Matthias Hüser and Xinrui Lyu for putting up with me,
Stefan Stark and David Kuo for their mastery of mythology and sports trivia,
Andre Kahles and Kjong Lehmann for sagely wisdom,
Cristóbal Esteban for understanding the struggle,
Francesco Locatello for giving me new perspectives,
Melanie Fernandez Pradier for lessons on work-life balance,
Martin Faltys for giving me the clinical side of the story,
and everyone else over the years, in the Rätschlab, Memorial Sloan Kettering Cancer Center, and the Institute for Machine Learning at ETH Zurich.

Outside of the Rätschlab, I have also been fortunate to benefit from the inimitable positivity of Charles Danko, and the guidance and wisdom of Danielle

Belgrave. I must also acknowledge my godmother Jennifer Pearson, the first person I met with a PhD, who made it look far too easy.

I would like to acknowledge the Tri-Institutional PhD Program in Computational Biology and Medicine, which started the whole affair of having confusingly many affiliations.

Crossing the Atlantic to live in a foreign land would have been a lot lonelier without evenings on the balcony at Coddington with Noah Dukler, Nate Tippins, Mary Centralla, Ozan Sener, Aditya Vaidyanathan, and everyone else in Ithaca.

Margie Hinonangan-Mendoza, Kadeem Ho Sang, Sabrina Nepozitek, and Natalia Marciniak went beyond the call of duty to smooth the complexities of doing a PhD across two continents.

Science cannot be done in an vacuum, and I am extremely grateful for the support of my friends over the last five years. You all brought me back to earth when I got carried away, and lifted my spirits in heavy moments.

In particular I am deeply indebted to the friendship of:
Jennifer Lorigan, who has always been there, and I hope always will be,
Neel S. Madhukar, whose exuberance and sincerity I shall always admire,
Natalie Davidson for her energetic creativity and kindness,
Faisal Alquaddoomi for his insightful and caring nature,
Cian Booth for his lightheartedness and for forcing me to take breaks,
Fionn Fitzmaurice for personifying freedom,
Regina Kelly and Ruairi Short for knowing what it's like to emigrate,
Matthew Carrigan for reminding me that not everything has to be rigorous, but everything must be rational,
The gestalt entity known as the Slackmind, for being a source of comfort and

debate in a fragmenting world,
Calvin Hu for liking all my tweets,
Zachary Elliott for patience during my second transatlantic voyage,
The members of NYC Resistor and New New York for giving me a third place for a while,
Zürich City Roller Derby for teaching me to be strong,
the rest of my friends scattered across the world,
and Oli Stiel, for dreaming with me.

Finally, I am endlessly grateful to my family; my dog Yuki, my sister Jessica and her husband Alex for forgiving my absentmindedness, and my parents Mary Walsh and Keith Hyland, for being unconditionally supportive and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

> Working in the hospital teaches you that there are only two kinds of people in the world: the sick and the not sick. If you are not sick, shut up and help.
>
> Hope Jahren, *Lab Girl*

1.1 Motivation

Computers have changed everything. The way we communicate, the way we work, the way we think about information. It is now possible to gather, transmit, and process data on a scale inconceivable to earlier generations. However, data on its own is meaningless. The challenge posed by the information age is precisely to convert this data into knowledge; to extract meaning from it, to make it useful. If we can do this, we can use it for the benefit of all.

This is especially true in medicine. Millions of people become sick every day. From every person's experience with illness, from every therapy they receive, every poor prognosis and remarkable recovery, lies the potential for medical breakthrough. If we could hear these patient stories, and understand the physiology and pathology that shaped them, the routinely-collected data of every-day practice would become a wealth of naturally-occurring experiments. Thanks to the increasing digitisation of healthcare data, in hospitals and by pa-

tients themselves, this is rapidly becoming a reality. As the information we collect increases in volume, and as advances in measurement technology present new ways of monitoring health and disease, we are approaching an era of data-driven medicine. In this new era, clinicians will be able to draw on the combined experiences of their peers to aid in decision making. They will work alongside intelligent systems which can review the entirety of a patient's health history in seconds. They will be supported by tools to help parse the ever-growing medical literature, to find the right clinical trial for their patient, to identify almost-imperceptible abnormalities they would previously have missed. But we are not there yet.

How do we go from data to knowledge? How do we identify what is important, what is a signal, and what is noise? We need computational tools to help make sense of data, to identify patterns as yet unknown. The closely-related disciplines of machine learning and statistics offer us the tools. Indeed, machine learning has a long history in medicine. Stretching back to the 70s, with the expert systems MYCIN [195] and INTERNIST-I [156], attempts have been made to encode medical knowledge in algorithms. However, while the modern era of machine learning has seen celebrated successes in computer vision, natural language processing, and recommender systems driven by deep learning, medicine is only beginning to benefit from these theoretical and computational advances. Medicine is a challenging domain. Mistakes cost lives. The data we collect comes from our patients' most vulnerable moments. The complexity of human physiology and its departures from homeostasis have occupied medical scientists for centuries. The questions we attempt to answer will not be solved overnight. The challenge to machine learning researchers is to find where its tools are needed, and conversely to draw inspiration from the challenges of

medical data to develop new methods to meet them.

One of these challenges is about *representation*. Much of machine learning is at its core about finding useful representations. Identifying spaces in which data is linearly separable, isolating invariant properties of data from noise attributes, and clustering high-dimensional objects into interpretable classes are all representation-learning problems. The goal of representation learning is to make structure hidden in data explicit. Representation learning in medicine is therefore about making sense of data. The processes of health and disease are dynamic, and the data we collect necessarily paints an imperfect picture. We see snapshots, irregularly and unreliably sampled, of physiological parameters of patients, and from this we ultimately seek to characterise their underlying state. Understanding how these measurements relate to each other, across data modalities and time, means constructing a computational representation of a patient.

This dissertation sits within an academic context of machine learning researchers, statisticians, health informatics experts, and clinicians who are working to realise the potential of data in medicine. It describes several steps towards that end, and outlines many more yet to be taken.

1.2 Structure of the dissertation

This dissertation is structured as a set of relatively independent chapters - thus each chapter has its own introduction and conclusion. These chapters explore distinct topics across representation learning in medical time series, and in several cases were done in conjunction with others. In such cases, individual contri-

butions are highlighted at the start of the chapter. Each chapter, its contribution and its motivation, is summarised briefly below:

- A large degree of healthcare data, especially that collected before EHRs became widespread, is stored in the form of written notes. Exploiting the information in these notes requires learning representations of *medical* language. Current state of the art in language representation learning relies on large text corpora, which are not easily obtained for medical text. In **Chapter Two**, I describe a method for learning representations of medical language which makes use of knowledge graphs in concert with free-text corpora to ameliorate this data-sparsity issue. i
- The potentially long-lasting impact of early health events necessitates representations of time series with a long memory. Retaining temporally distant information is a known challenge for recurrent neural networks, which are one of the most common time-series modelling approaches in machine learning. In **Chapter Three** I present a novel and efficient parametrisation of a unitary recurrent neural network using the correspondence between Lie group and Lie algebra, and demonstrate how such a model is capable of retaining information over long time horizons.
- Sharing medical data is challenging due to its private and sensitive nature. This poses a challenge for machine learning researchers in healthcare, where work is often non-reproducible and non-comparable due to the use of private datasets. In **Chapter Four**, I describe a method to *generate* synthetic medical time-series using generative adversarial networks, as well as a novel evaluation method for synthetic data and an analysis of its privacy properties.

- **Chapter Five** presents another perspective on representation learning in healthcare. In this final chapter, I describe an early warning system for circulatory system deterioration in intensive care patients. This translational setting demonstrates the need for eminently usable and interpretable representations of time-series, which we obtain with engineered features based on clinical prior knowledge. Building such a warning system underscores several challenges in the practical application of machine learning in medical time-series classification, such as data quality control, task specification, and evaluation.

CHAPTER 2

MEDICAL TEXT

> A model is a lie that helps you see the truth
>
> Siddhartha Mukherjee
>
> *The Emperor of All Maladies*

Individual contributions *The work in this chapter was done in collaboration with Theofanis Karaletsos and my supervisor Gunnar Rätsch, who provided guidance and ideas, and helped to write manuscripts resulting from the work. Everything else, including implementation and experiments, was performed by me. This work was published at the 30th AAAI Conference in Artificial Intelligence in 2016 [102], and at the workshop on Machine Learning for Healthcare at NIPS 2016 [101].*

Natural language is both ubiquitous in human activities and challenging to analyse computationally. While abstraction layers in other systems have enabled automated processes not reliant on natural language, this is not the case for many aspects of healthcare. The use of language to deliver instructions, to make records, and to transfer information is commonplace. This is likely in part related to the challenges associated with adopting new technologies in risk-critical domains, but it is also a feature of processes which include human-to-human interactions. We don't expect doctors to communicate to their patients or each other in machine code, nor should we. To computationally exploit these communications we therefore need methods to analyse and extract information from natural language. We do this by learning representations of medical

language by embedding words, concepts, and relationships between them in a vector space.

Processing medical text poses several particular challenges:

1. Medical text corpora lack the abundance and scale of generic English corpora, such as the Gigaword news dataset [173].
2. Medical text notes are often hastily written, with abbreviations and sentence fragments.
3. Medical language contains many specific and highly technical terms, and implies certain senses of words which may be uncommon, for example 'patient' would often be a noun in a medical context, and an adjective elsewhere.

The last two points highlights that direct application of methods developed on standard English to medical English can fail. Wang et al. [228] demonstrate that word representations learned from a medical corpus outperform generic embeddings in semantic relatedness tasks for medical concepts, and Denecke and Deng [49] highlight the need for domain-dependent models in sentiment analysis. The first point motivates the need, explored in this chapter, to combine information from sources beyond text corpora to learn medical text representations. Especially given the long tail of rarely-used and technical words found in medical language, augmenting limited text corpora with information from an ontology, such as the UMLS [20], enables the learning of higher-quality concept representations. Despite the challenges associated with the domain, the use of text representations in medical natural language processing is a growing field. Jonnalagadda et al. [111] use vector representations of words, learned form a

large unannotated medical text corpus, to enhance clinical concept extraction from text notes. Krompaß et al. [127] learn latent representations of diagnosis and procedure codes to predict hospital readmission. Ghassemi et al. [70] represent clinical notes using topic models to predict mortality in intensive care patients, while Grnarova et al. [79] later address the same task with representations derived from deep networks. Miotto et al. [157] also use topic modelling to represent text alongside other electronic health record features to predict future disease status. Very recent work [12] has exploits a large graph of co-occurrences built from clinical records [63], to learn embeddings for thousands of medical concepts and provide them to the research community.

This chapter is laid out as follows: Section 2.1 provides background and context for learning semantic representations of words. Section 2.2 describes how facts about the world are represented in knowledge graphs, and how these graphs are constructed. In Section 2.3 the model proposed in this chapter and related contributions are described. This model is analysed in several settings using both medical and non-medical language in Section 2.4, and Section 2.5 summarises the chapter and outlines questions for future work.

2.1 Word representations

Language is composed of a hierarchy of elements - words become phrases, which become sentences and paragraphs, and so on. While sub-word level components have been studied in the representation-learning literature [240] (for example characters [119], morphemes [143], radicals [206]), in this work we focus on the most readily-discretised unit of language in English: the word.

In some cases, we consider medical concepts which comprise several English words, such as 'lung cancer', but we represent each concept as a discrete token.

Given word representations, it is possible to form phrase [18], sentence [122], or document [130] embeddings. Embeddings can be used in downstream tasks such as named-entity recognition [175], document classification [80], sentiment analysis [52], neural machine translation [180], and mixed-modality applications such as zero-shot learning learning in computer vision [198] and video-text alignment [21].

One-hot encoding

The simplest representation of categorical-valued data like words is the one-hot encoding. Given a dictionary of N words, we can form a simple representation of any word using this encoding; the ith word in the dictionary is a N-dimensional vector of zeroes,with a 1 in the ith position;

$$\mathbf{z}_i = [0, \cdots, 1, \cdots, 0] \tag{2.1}$$

This enables us to further represent sentences or documents by summing or averaging over the one-hot-encoded words they contain - this represents the document as a 'bag of words', discarding order and locality.

The one-hot encoding is perhaps the simplest representation available for discrete concepts, but it implies an unrealistic independence assumption: all words are equally similar (or dissimilar) to all other words. This is clearly/obviously/evidently false.

As well as producing a representation which doesn't capture expected semantics, the one-hot encoding produces N-dimensional vectors for each word, where N is invariably large - 600,000 unique words is not uncommon. Acknowledging that many of these N are liable to be synonyms or antonyms, this means naive models using such inputs will be overparametrised and prone to overfitting, assuming training is even practical with so many parameters. This leads us to seek representations which are lower-dimensional (than N), inhabiting a space equipped with a similarity measure which reflects some level of *semantic* similarity.

2.1.1 Distributional semantics

Before we can devise an algorithm to learn word representations from data, we need to make concrete the desired notion of *similarity*. When we say that two words are *similar*, we might then clarify to say that the words have similar meanings - they are semantically similar. So how do we go about determining semantic similarity from data? This is question is the objective of the field of statistical semantics, and it broadly relies on an assumption known as the *distributional hypothesis*. The distributional hypothesis was first described by Harris in a 1954 paper [82]. In this original work, Harris proposed that language can be described in terms of its distributional structure, independent of history or meaning. The distributional structure of a language is defined to be the occurrence of parts in relation to other parts, for example using co-occurrence statistics (although the technology to compute such statistics from large corpora was not available at the time). The 'parts' in question could refer to elements of the language at different levels, for example phonemes, morphemes, or words. The

importance of the distributional hypothesis for *meaning* is seen in the following quote, from Harris [82]; *'if we consider words or morphemes A and B to be more different in meaning than A and C, then we will often find that the distributions of A and B are more different than the distributions of A and C'*.

In other words, difference of meaning correlates with difference of distribution.

This was later restated more elegantly by Firth in 1957 [64]; Citing Wittgenstein's *'the meaning of words lies in their use'*, he said *'You shall know a word by the company it keeps'*. The distributional hypothesis therefore states [188] that *'Words which are similar in meaning occur in similar contexts'*.

This points conveniently to an unsupervised approach to representation learning on text: identify words sharing a context, for some notion of 'context'.

Types of meaning

The previous observation about syntagmatic and paradigmatic relationships raises an important subtlety in the pursuit of word representations. In Sahlgren [188], they argue that although distributional approaches are bound to capture broad notions of similarity (including syntagmatic and paradigmatic senses), this nonspecifically-defined notion of similarity is nonetheless meaningful. However, in specific instances it is clear that context plays an important role in determining similarity. Distributional approaches seek to align words with those sharing similar contexts *in general*. Under a distributional approach, synonyms should logically be assigned similar representations, but it is unclear how to treat words in other relationships. Should the antonym for word j be

mapped to $\mathbf{z}_j$, or to some vector orthogonal to $\mathbf{z}_j$? This question is not possible to answer in general, because whether a word is similar or dissimilar to its antonym is context-dependent. Are 'happy' and 'sad' similar because they are emotions, or dissimilar because they imply opposing sentiments? Should 'happy' be instead mapped closer to 'safe' (a word labelled with positive sentiment [97]).

The hope for representation-learning is that the many *aspects* of the word's meaning are captured simultaneously in the embedding. This implies that there exist subspaces in the embedding space which correspond to different notions of similarity. If these subspaces exist and are linear, then combining element-wise distances in the embedding space to form an overall measure of pairwise (dis)similarity makes intuitive sense, justifying a Euclidean metric on the space.

These and related questions have prompted computational scientists to devise methods for extracting this semantic information from data. The next sections describe several approaches.

2.1.2 Latent semantic indexing

Traditionally, one can represent distributional information by constructing a sparse matrix M of words versus possible contexts. In its simplest form, the entry M_{ij} is the number of times word i appeared in context j.

In information retrieval, the context of interest is a *document*, thus the matrix becomes a term-document matrix. The distributional hypothesis then states that terms appearing in the same documents (similar contexts) must be themselves

similar.

The idea behind latent semantic indexing[48] is to perform a reduced-rank singular-value decomposition (SVD) on the term-document matrix.

$$M = U\Sigma V^T \tag{2.2}$$

where U and V^T are orthogonal matrices and Σ is diagonal, consisting of the eigenvalues of M. Restricting Σ to the top k eigenvalues, Σ_k, we then have $M_k = U_k\Sigma_k V_k^T$. It is possible to obtain k-dimensional representations of both terms *and* documents for the purpose of comparison through:

$$\begin{aligned} \mathbf{t}_a &= (U_k\Sigma_k)_a \\ \mathbf{d}_b &= (\Sigma_k V_k^T)_b \end{aligned} \tag{2.3}$$

where $\mathbf{t}_a$ is the representation of term a, and $\mathbf{d}_b$ is the document b, and both are elements of $\mathbb{R}^k$.

Rather than using document co-occurrence to build this matrix, an alternative approach is to consider words in the context of each other. In this setting, the co-occurrence matrix is square, representing all pairwise sets of words. Word i is considered to be in the context of word j if they appear within a *context window*. This context window could be directed, for example $M_{ij} = 1$ if word j appeared once within c words *after* word i, but it is typically taken to be symmetric. That is, M_{ij} counts the number of times j appeared within c words of word i, and vice versa.

2.1.3 Neural word embeddings

The use of neural networks for obtaining distributed word representations was motivated by the curse of dimensionality in language modelling. Bengio et al. [15] propose the use of distributed word representations to aid in modelling the conditional probability of words given a preceding sequence;

$$p(w_t|w_1, \cdots, w_{t-1}) \tag{2.4}$$

Such conditional distributions appear frequently in Markov modelling of n-grams, where they can be computed explicitly from a large corpus if n is small. However, as n grows larger (beyond 2 or 3), the number of possible n-grams grows faster than the size of available corpora, resulting in unobserved events, and prohibitively large look-up tables This imposes a restriction on either the length of modelled sequences (n), or the size of the vocabulary. This is specific instance of the curse of dimensionality which Bengio et al. attempt to overcome. As well as suffering from computational limitations, approaches based on n-grams needlessly ignore *word similarity*. If one could instead model the conditional probability based on semantic similarity, it would be possible to make predictions about unseen sequences of words, provided they could be represented in the embedding space. This is the idea presented in Bengio et al. [15] and elaborated on by subsequent work.

Explicitly, they parametrise the conditional probability of Equation 2.4 as follows:

$$p(w_t|w_1, \cdots, w_{t-1}) = \frac{\exp(y_{w_t})}{\sum \exp(y_i)} \tag{2.5}$$

where y_i is the unnormalised probability of observing word i, and is given by a

standard neural transformation:

$$\mathbf{y} = \mathbf{b} + W\mathbf{x} + U \tanh(\mathbf{d} + H\mathbf{x}) \tag{2.6}$$

Here $\mathbf{b}$, W, U, $\mathbf{d}$, and H are parameters of the model, and $\mathbf{x}$ is the concatenation of the *representations* of the words in the sequence.

$$\mathbf{x} = [\mathbf{z}_1, \cdots, \mathbf{z}_{t-1}] \tag{2.7}$$

We will refer to the parameters $\mathbf{z}_i$ corresponding to the representation of word i as the *embedding* of word i. These $\mathbf{z}_i$s are effectively parameters of the model, they can be thought of as rows in a $|V| \times d$-dimensional matrix, where d is the dimensionality of the embedding space. These parameters are understood to be the coordinates of word i in some d-dimensional Euclidean space, hence word i has been *embedded* in $\mathbb{R}^d$.

Since the publication of this work in 2003, thousands of follow-up papers have been published, elaborating and refining the neural word embedding approach. One particularly influential example is the `word2vec` model proposed by Google researchers in 2013 [152, 153]. Since this model forms the inspiration of the novel method described in this chapter, it is described in detail in the next section.

2.1.4 Skip-gram and continuous-bag-of-words

The seminal neural language model of Bengio et al. [15] learns word representations by predicting the *next* word in a sequence. Given a set of T words, the objective is to predict word $T + 1$ - this is the same philosophy employed by predictive n-gram models, building on Shannon's observations about the predictability of English[192]. While it is demonstrably true that n-gram and neural

models succeed at their task of predicting words, the representations learned (if available) are naturally optimised for this task. To learn a general-purpose representation of a word in terms of its meaning, we can exploit distributional semantics (described in Section 2.1.1). Predictive models such as that of Bengio *et al.* arguably exploit distributional information, since a word must be predicted from the context of the preceding sentence. Explicitly distributional models extend this notion of context to include a potentially symmetric window around the word.

One such explicitly distributional model is `word2vec` [153]. This model refers to two slightly different ideas which both exploit the distributional hypothesis:

- The `skip-gram` model attempts to predict the words neighbouring a given word w_t.
- The `CBOW` (continuous bag-of-words) model attempts to predict the word w_t given the average representation of its neighbours.

Explicitly for skip-gram, the objective is to maximise the log-probability of the observed words under a conditional model;

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.8}$$

where c is the number of words before and after w_t which comprise its context.

This conditional probability is defined using a softmax and introduces the embeddings as parameters:

$$p(w_{t+j}|w_t) = \frac{\exp\left(\tilde{\mathbf{z}}_{t+j}^T \mathbf{z}_t\right)}{\sum_i \exp\left(\tilde{\mathbf{z}}_i^T \mathbf{z}_t\right)} \tag{2.9}$$

Here $\mathbf{z}_j \in \mathbb{R}^d$ is the embedding of word j.

They distinguish between the embedding of the central word ($\mathbf{z}_t$) and those of the surrounding context words ($\tilde{\mathbf{z}}_{\mathbf{t+j}}$), thus giving each word *two* embeddings (both live in $\mathbb{R}^d$). Goldberg and Levy [72] provide a motivation for using two representations for each word. They say, '*One motivation for making this assumption is the following: consider the case where both the word dog and the context dog share the same vector* $\mathbf{v}$*. Words hardly appear in the contexts of themselves, and so the model should assign a low probability to* $p(dog|dog)$*, which entails assigning a low value to* $\mathbf{v}^T\mathbf{v}$ *which is impossible.*'

We can clarify this intuition by observing that words with similar $\mathbf{z}$ representations share a *paradigmatic* relationship in that they may be exchangeable in sentences, but do not tend to co-occur. For example, 'dog' and 'puppy' are in a paradigmatic relationship, and we would consider an embedding model successful if it places $\mathbf{z}_{\text{dog}}$ somewhere near $\mathbf{z}_{\text{puppy}}$. Conversely, words s and t with $\mathbf{c}_s \approx \mathbf{v}_t$ have a *syntagmatic* relationship and tend to co-occur (e.g. Sahlgren [188]). For example, 'dog' and 'puppy' are both in syntagmatic relationships with 'bark', so $\mathbf{z}_{\text{dog}} \approx \mathbf{c}_{\text{bark}}$ and $\mathbf{z}_{\text{puppy}} \approx \mathbf{c}_{\text{bark}} \rightarrow \mathbf{z}_{\text{dog}} \approx \mathbf{z}_{\text{puppy}}$. That is, we seek to enforce syntagmatic relationships and through transitivity obtain paradigmatic relationships of $\mathbf{v}$ vectors.

The skip-gram model (and related continuous-bag-of-words model) therefore exploits the distributional hypothesis by requiring that words resemble their context, leading to words with similar contexts having similar representations. Cosine similarity is used to assess the similarity between words;

$$s(\mathbf{z}_a, \mathbf{z}_b) = \frac{\mathbf{z}_a^T \mathbf{z}_b}{\|\mathbf{z}_a\| \|\mathbf{z}_b\|} \tag{2.10}$$

Evidently this is equivalent to the inner product when word vectors are nor-

malised to unit length.

Learning the embeddings in the `word2vec` model ostensibly requires optimisation of the objective in Equation 2.8 where the conditional probability is given by the softmax in Equation 2.9. However, this representation for the conditional probability is intractable due to the normalisation term, which requires summing over the whole vocabulary. To address this, Mikolov et al. [153] propose two alternative objectives: the hierarchical softmax and negative sampling. Thus, given the choice between skip-gram and CBOW, and the choice of modified objective, the phrase '`word2vec` model' can refer to one of four models. These choices are essentially hyperparameters of the model, with the authors observing that hierarchical softmax performs better for infrequent words, while negative sampling is superior for frequent words and lower-dimensional embeddings [5].

Negative sampling replaces the conditional probability (assuming the skip-gram model as in Equation 2.9) with the following expression:

$$\log \sigma \left(\mathbf{z}_{t+j}^T \tilde{\mathbf{z}}_t \right) + \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma \left(-\mathbf{z}_{w_i}^T \tilde{\mathbf{z}}_t \right) \right] \tag{2.11}$$

In practice, the expectation value is replaced by an estimate. They define the noise distribution $P_n(w)$ as $P_n(w) \propto U(w)^{3/4}$, where $U(w)$ is the unigram distribution over words.

Goldberg and Levy [72] demonstrate that this objective can be derived by defining a conditional distribution for whether a pair of words w_i, w_j appear together in a context in the dataset;

$$p(D = 1 | w_i, w_j; \theta) = \sigma(\mathbf{z}_i^T \tilde{\mathbf{z}}_j) \tag{2.12}$$

It would then in principle be possible to learn θ by maximising the likelihood of

the observed data. However, this would result in useless embeddings where all vectors are identical, since there is no cost for making $\mathbf{z}_i \sim \mathbf{z}_j$ if w_i and w_j are *not* observed together. To resolve this, Goldberg and Levy [72] demonstrate that it is sufficient to introduce some synthetic *noise* pairs w_a, w_b such that $p(D = 1|w_a, w_b; \theta)$ should be low. Using the noise distribution $P_n(w) \propto U(w)^{3/4}$ to generate the noise pairs, this recovers the essence of Equation 2.11. Interestingly, it has also been shown that the `word2vec` model is equivalent to matrix factorisation for special choices of the term-term matrix [134].

The model we introduce in Section 2.3 is a generalisation of the *idealised* version of skip-gram `word2vec` - the one described by Equations 2.8 and 2.9. We retain the probabilistic interpretation by avoiding explicitly computing the normalisation factor using persistent contrastive divergence, detailed in Section 2.3.3.

2.2 Representing facts

The previous section investigated how *distributional* information can encode word meaning - how meaning can be inferred from the use of the word, represented by the statistics of its neighbouring context. Another way to represent word meaning is to do so explicitly in the form of statements such as `a dog is a mammal`, or `lungs contain bronchi`. Individual such statements naturally only capture part of a word's meaning, but a large set would produce an increasingly comprehensive view. This chapter therefore focuses on how and why facts like these are represented, how they're learned, and finally how this view can be combined with the distributional approach of the previous chapter.

2.2.1 Knowledge graphs

Organising knowledge by way of categorising and relating concepts to each other is an ancient human endeavour. These categorisations of knowledge are known as ontologies. An ontology consists of a set of possible entities and relationships which exist between them, for example statements like 'Serena Williams was born in Michigan' would belong to an ontology where people and places are entities, and relationships such as 'born in', 'lives in', 'neighbours', and so on. We see that some relationships may be invalid for some pairs of entities - for example, the statement 'Michigan was born in Ohio' is not a meaningful statement (unless 'Michigan' is also a person). Ontologies can therefore also contain information about which types of statements are valid, for example by including categorical information and only allowing certain relationships between entities from certain categories (like `Person` and `Place`). If we consider entities to be nodes in a graph, and relationships are directed edges, then we can represent an ontology as a *knowledge graph*. An ontology equipped with categorical information could additionally feature a second graph, containing categories as nodes and valid relationships between them as (directed) edges.

Beyond the intrinsic value of constructing ontologies, knowledge graphs can represent facts about the world (or a specific domain or system) in a format amenable to computation. This means knowledge graphs can be immensely useful for information retrieval, fact-answering, disambiguation, or potentially any application which exploits facts about the world. For example, Google search is backed by the 'Google Knowledge Graph' [19], which is based on the Freebase knowledge base [22], as well as Wikipedia and the CIA World Factbook. The DeepQA system behind IBM's Watson [62] known for winning Jeop-

ardy! in 2011, also exploits several knowledge graphs such as dbPedia [133], WordNet [61], and Yago [205]. A system called NELL [159]- the Never-Ending Language Learner, developed at Carnegie-Mellon University, has been running since 2010, 'reading' the internet and extracting facts like `pesos` are the `currency of Chile` and constructing an openly-accessible knowledge base. More recently, the ANGELINA system for automatically creating games [44] is capable of crowd-sourcing facts from Twitter by asking questions like 'Would it make sense for a cow to move?'

In this chapter, we focus on two knowledge graphs: WordNet [61] and SemMedDB [118], which are described in detail in Section 2.4.1.

2.2.2 Building and representing knowledge graphs

While knowledge graphs are highly valuable, their utility comes at a cost. High-quality databases typically require human curation or annotation. Projects like NELL aim to automatically generate entries in a knowledge graph, and use crowdsourced curation to prune 'unreliable' entries or collect true facts. One approach to building useful knowledge graphs is to begin with a manually curated set, and then *grow* the graph. For example, since the relationship 'located in' is transitive, it is possible to add new edges. Using the earlier example, since 'Michigan' is located in 'USA', the fact that 'Serena Williams was born in Michigan' can lead us to conclude (correctly) that Serena Williams was born in the USA, and add an appropriate edge. Moreover, equipped with the knowledge that citizenship in the USA is a birthright (at the time of writing), we can also conclude that 'Serena Williams is a citizen of the USA'. While this approach to

growing knowledge graphs is conceptually simple and potentially highly accurate, it is limited both in use and impact. Expanding a graph in this way requires knowledge of how one relationship may imply others (as in the case of 'born in' $\rightarrow$ 'citizen of'), necessitating some information about the entities in the ontology (an example being the Semantic Network [145], which accompanies the UMLS Metathesaurus [20]). It is also limited to adding new *edges* alone, as facts about new entities cannot be logically inferred in this manner.

Rather than using logical rules to extend the graph, one can use *statistical relational learning* to try to predict new elements of the graph using statistical models. In general, probabilistic models can be used to represent relational data of the form 'entity i is related to entity k through relationship j' by associating this statement with a binary random variable y_{ijk} whose value denotes the truth of the statement. One could then parametrise this model using latent variables x_{ijk} (following the notation of Nickel et al. [168]) for each triple (i, j, k). For example, we could have

$$y_{ijk} \sim \text{Bernoulli}(x_{ijk}) \tag{2.13}$$

Modelling the knowledge base would then amount to finding the x_{ijk} to maximise the joint likelihood of the observed data, assuming all y are conditionally independent given all x. This simple model as described is of course useless and computationally impractical, as it learns one latent variable (x_{ijk}) for each possible triple, and cannot generalise to unobserved entities. To make such latent variable models practically useful for knowledge base completion or entity embedding, structural assumptions about the latent variables are made. That is, rather than representing each triple (i, j, k) by an independent latent variable, we can represent i, j, k independently and combine them into an overall representation of the triple. Much work in statistical relational learning has thus

focused on how to represent the components, how to combine them, and how to use this to parametrise the probabilistic model. The work in this chapter sits within this framework, and we review some notable related work here.

RESCAL

Relational data lends itself naturally to representation using a third-order tensor. Tensor factorisation can then be used to find lower-dimensional representations of tensor components, and thus individual entities. This is what is done in the RESCAL model[167]. In RESCAL, the tensor $\mathcal{X}$ represents the knowledge graph, such that $\mathcal{X}_{ijk} = 1$ indicates that the triple (i, k, j) is true. For non-existent (false) or unknown triples, the entry is set to 0. The choice to equate 'unknown' with 'false' is referred to in the literature as the closed-world assumption - in this paradigm, all unobserved facts are assumed to be false.

In RESCAL, the tensor is sliced along its final dimension (the relationship or predicate dimension) and each slice $\mathcal{X}_k$ is factorised as follows:

$$\mathcal{X}_k \simeq AR_kA^T \tag{2.14}$$

where A is a $n \times r$ matrix consisting of the representations of the n entities, and R_k is a $r \times r$ matrix modelling their relationships under the k-th relationship. R_k is in general not symmetric, allowing for directed relationships between entities. By examining a single component of $\mathcal{X}_{ijk}$ under the approximation, we see that RESCAL learns a single r-dimensional representation for each entity:

$$\mathcal{X}_{ijk} \simeq \sum_{\alpha\beta} A_{i\alpha} R_k^{\alpha\beta} (A^T)_{\beta j} = \mathbf{a}_i^T R_k \mathbf{a}_j \tag{2.15}$$

where $\mathbf{a}_i$ is the i-th row of A. While RESCAL is not explicitly probabilistic, it can be understood in the context of Equation 2.13 by interpreting $\mathcal{X}$ as the tensor of

latent factors x_{ijk} in the limit where observed triples have probability 1 under the model.

New links can be predicted under the RESCAL model by computing the value of the $\mathcal{X}$ tensor for the queried link, e.g. $\mathcal{X}_{lmn}$, and asking if it exceeds some threshold θ. Similarly, for a pair of entities (l, m), the type of relationship between them can be predicted by comparing $\mathcal{X}_{lmn}$ for all choices of n.

Neural Tensor Network

Socher et al. [197] describes an approach using neural networks to represent facts in knowledge bases. Their model, called the Neural Tensor Network (NTN) takes as inputs two concepts and a relationship, and outputs a confidence score that this triplet is true. For example, the question `Does a Bengal tiger have a tail?` can be answered by feeding in the representations for `Bengal tiger` and `tail`, into a function (a neural network) indexed by the relationship `has part`. More explicitly, it produces certainty g for the truth of the triple (i, j, k) (following the notation of the previous sections) as follows:

$$g(i,j,k) = \mathbf{u}_j^T \tanh\left(\mathbf{e}_i^T W_j \mathbf{e}_k + V_j \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_k \end{bmatrix} + \mathbf{b}_j\right) \tag{2.16}$$

where $tanh$ is applied element-wise. Therefore, this model is called the neural *tensor* network, because W_j is a rank-3 tensor with dimensions (d, m, d) such that the product with the embeddings $\mathbf{e}$ produces a vector in $\mathbb{R}^m$. Similarly, V_j has shape $(m, 2d)$, and $\mathbf{b}_j$ and $\mathbf{u}_j$ are also in $\mathbb{R}^m$. Thus, each relationship j has its own representation in terms of $\mathbf{u}_j$, W_j, V_j, $\mathbf{b}_j$.

Collecting all the parameters of this model, including the embeddings $\mathbf{e}$, into

Ω, they are obtained by minising the following objective:

$$J(\Omega) = \sum_{i=n}^{N} \sum_{c=1}^{C} \max\left(0, 1 - \left(g(i_n, j_n, k_n) - g(i_n, j_n, \tilde{k}_n^c)\right)\right) + \lambda \|\Omega\|_2^2 \qquad (2.17)$$

here, the triplet (i_n, j_n, k_n) is a true example from the dataset, of which there are N. The c index tracks *corrupted* triplets - for each real triple, they generate C of these. A corrupted triple corresponding to the nth training example contains the same first entity (i) and relationship (j) as the true triple, but where the corresponding entity (k) has been replaced with a random one, which is denoted here as $\tilde{k}_n^c$. Therefore, the model has to correctly assign a high confidence g to the real triple, and a low confidence to the false one. To avoid the task being trivial, the corrupted entity $\tilde{k}_n^c$ is chosen such that it can appear in that position with that relationship, but not with the other entity. Put another way, $(\tilde{i}, j_n, k_n^c)$ is a true triplet for some $\tilde{i} \neq i_n$.

Trans*

One of the oft-cited properties of the `word2vec` approach is the (seemingly unintentional) tendency for the embedding space to allow for analogical reasoning using vector arithmetic. An example is a consistent offset vector existing between the representations for 'Berlin' and 'Germany', and 'Dublin' and 'Ireland', seemingly representing the relationship `capital city of country`. Coupled with this, and the fact that hierarchical relationships naturally lend themselves to representation through translation, the TransE [25] model differs from RESCAL and other bilinear approaches in representing relationships as *translation vectors*.

In TransE, the intuition is that if i and j are related through k, then $\mathbf{e}_i + \mathbf{l}_k$

should be similar to $\mathbf{e}_j$. To quantify similarity they use a dissimilarity measure d such as the Euclidean distance, which is equivalent up to constants to the inner product for vectors with fixed norm. Once again augmenting the data with a set of negative examples (generated by corrupting true triples), the optimisation problem is to rank the true examples more highly. Specifically they maximise

$$\mathcal{L} = \sum_{(i,j,k)\texttt{true}} \sum_{(i',j',k)\texttt{false}} [\gamma + d(\mathbf{e}_i + \mathbf{l}_k, \mathbf{e}_j) - d(\mathbf{e}_{i'} + \mathbf{l}_k, \mathbf{e}_{j'})]_+ \tag{2.18}$$

While the TransE approach is simple, various extensions have been proposed, such as TransH [229], TransM [58], and TransR [136].

2.2.3 Combining knowledge graphs with free text corpora

While one of the contributions of the model described in this chapter is its ability to exploit both distributional and graph-based knowledge, this is not the first time data from these sources have been combined. This idea was already explored in Weston et al. [233] for example, where entities belonging to a structured database are identified in unstructured (free) text in order to obtain embeddings useful for relation prediction. Similar approaches are also employed by Fried and Duh, Wang et al., Xu et al., Yu and Dredze [66, 230, 236, 242]. In these cases, separate objectives are used to incorporate different data sources, combining (in the case of Xu et al. [236]) the skip-gram objective from Mikolov et al. [153] and the `TransE` objective of Bordes et al. [25]. The method proposed in this chapter (Section 2.3) uses a single energy function over the joint space of word pairs with relationships, combining the 'distributional objective' with that of relational data by considering free-text co-occurrences as another type of relationship.

Rather than learning both simultaneously, it is also possible to modify existing word embeddings, trained using distributional sematics, to adhere to a given knowledge base or ontology. This is done for example in Faruqui et al. [59] using semantic lexicons on top of GloVe vectors [175], Johansson and Nieto Piña [108] use SALDO [27] (an alternative to WordNet), and Mrksic et al. [163] use antonmy and synonymy relationships to retrofit word embeddings in a similar way. Jauhar et al. [105] explore both retrofitting embeddings and modifying the objective of the `word2vec` model [153] to respect word sense encoded in WordNet. The retrofitting approach of Faruqui et al. [59] has also been applied to medical text [243]. The paradigm of first learning word representations from free text, and then modifying them using information from a knowledge base (such as a semantic lexicon) could be considered a special case of the joint approach taken in this chapter, corresponding to a curriculum of training first on the unstructured data source.

2.3 Contributions

This chapter describes a contribution to the field of natural language processing in the form of a representation-learning approach for words. Inspired by the challenges of medical language, where we would like to supplement scarce corpora of free text with information from manually curated knowledge graphs, our contribution is thus a model with the following properties:

1. The result of a trained model is a set of word embeddings using the distributional framework, as well as matrix-valued representations for relationships between words

2. The model can be trained on free and structured text corpora to learn the above
3. The model describes a joint probabilistic of words and relationships, which enables prediction and completion of `word-relationship-word` triples, allowing for knowledge graph extension and other queries

This section describes how this is achieved. We then examine the model in several settings, and demonstrate its use in medical language processing for knowledge graph completion and knowledge transfer (Section 2.4.5).

2.3.1 Learning representations of words and relationships

We assume we have a vocabulary of V words, and R possible relationships which can potentially hold between them, for example `willow` is a `type of tree`.

Thus, our focus is on triplets of the form (S, R, T) where S and T refer to elements of the vocabulary (implying 'source' and 'target'), and R is the relationship between them. We then seek *vector representations* for each word in V *in its role* as either source or target, and *matrix representations* for each relationship. Following the approach of a neural language model, we learn these representations in an 'encoding' layer - that is, we cast representation-learning as learning the parameters of a model. Specifically, we perform stochastic maximum likelihood on a probabilistic model defined over the triplet of random variables (S, R, T). Inference is outlined in Section 2.3.3. First, we describe the model.

2.3.2 Relationships as context

Our approach uses the fact that skip-gram-type models only rely on similarity *between* word representations. This means that we can *implicitly* capture polysemy and homonymy by allowing for different types of pairwise similarity. For example, the word 'bank' likely refers to an establishment if it appears in a statement such as 'banks contain rooms', while its use in bounding rivers is apparent from a statement like 'she slid down the bank'. A riverbank is unlikely to contain, employ, contact, or otherwise actively interact with a person, whereas a banking institution is of little interest to wildlife and unlikely to have a river burst from it. However, if we give the token 'bank' a single vector representation, should it be closer to 'credit union' or to 'shore'?

We address this polysemy issue by defining a set of possible relationships between words in our dictionary. Each relationship implies a transformation of the embedding space, which produces a *variant* of a word's embedding, corresponding to its meaning *in the context of* the relationship. Specifically, we define it as an affine transformation of the embedding space:

$$R : \mathbb{R}^d \to \mathbb{R}^d \tag{2.19}$$

The affine transformation corresponding to relationship r is represented using an augmented matrix:

$$R_r = \left(\begin{array}{ccc|c} & A & & \mathbf{b} \\ 0 & \cdots & 0 & 1 \end{array}\right) \tag{2.20}$$

where A is a $d \times d$ matrix and $\mathbf{b}$ is a d-dimensional vector. Hence, R_r is a square, $d+1$-dimensional matrix. The action of R_r is to map the context vector $\tilde{\mathbf{z}}_j$ for word j to another point in $\mathbb{R}^d$ corresponding to the representation of word j in the context of relationship r.

The choice of an *affine* transformation is motivated by the observation [153] that certain relationships are already captured as *translations* by relationship-agnostic word embedding models, which was discussed in Section 2.2.2. For this augmented representation to work, it is necessary to append a fixed 1 on to all word vectors, which otherwise does not influence their representation.

One can extend the softmax expression used in the `word2vec` model with this modification;

$$p(w_{t+j}|w_t, r) = \frac{\exp\left(\tilde{\mathbf{z}}_{t+j}^T R_r \mathbf{z}_t\right)}{\sum_i \exp\left(\tilde{\mathbf{z}}_i^T R_r \mathbf{z}_t\right)} \tag{2.21}$$

and it is then clear that this reduces to the original case when $R_r = \mathbb{I}$.

2.3.3 Probabilistic modelling

While many language models represent conditional probabilities, that is,

$$p(w_t|w_{t-1}, \cdots, w_0) = f_\theta(w_{t-1}, \cdots, w_0) \tag{2.22}$$

where f_θ is some parametrised function (such as a recurrent neural network) which is learned through optimisation of a loss such as cross-entropy, in this work we rather model the *joint* probability of the observed data.

Recalling as before that we consider triplets - (S, R, T) (source, relationship, target) - this is a generalisation of the case in word2vec where only target and context (=source) are considered - we seek to define the probability of observing such a triplet - $P(S, R, T)$. Note that while we refer to 'words', S and T could represent any entity between which a relationship may hold without altering our mathematical formulation, and so could refer to multiple-word entities (such as `UMLS` Concept Unique Identifiers) or even non-lexical objects. With-

out loss of generality, we refer to them as words.

If we suppose we can parametrise this probability distribution, then the parameters can be learned through, e.g. maximum-likelihood estimation given an observed set of triplets (a dataset). The advantage of having the joint distribution $P(S, R, T)$ is that it gives us immediate access to other quantities of interest through:

1. Marginalisation: given $P(S, R, T)$ we can get any marginal quantity, e.g. $P(s, t) = \sum_r^R P(s, r, t)$, which would give the probability of observing words s and t in *any* relationship together.
2. Conditioning: given $P(S, R, T)$ we can also ask for conditionals, e.g. given we observe word s and relationship r, what words t are most likely?

$$P(R|s, t) = \frac{P(s, R, t)}{\sum_r^R P(s, r, t)} \tag{2.23}$$

(and equivalently we can get any other conditional distribution)

Note that we use the convention of capital letters to refer to random variables, and lower case to refer to specific values they take - so S refers to the random variable whose range is all words which can appear as the 'source' in a triple, and s refers to a particular instance of S. These quantities are not readily obtained by other methods.

A standard method to parametrise a probabilistic model is to use a Boltzmann distribution,

$$P(\Omega) = \frac{e^{-\mathcal{E}(\Omega)}}{Z} \tag{2.24}$$

here $\mathcal{E}(\omega)$ is the energy of the state ω, and Z is the partition function which ensures the probability density is normalised.

Energy function

We define consider a state in this system to be a triple S, R, T, indicating that concepts S is related to T through R. This means the state-space is the set of all possible triples - a discrete space of size $|S||R||T|$.

Since we are looking to learn representations of words, we need to parametrise the energy function in terms of these representations. Every word, indexed by s say, is mapped to a point in d-dimensional Euclidean space[1], $w_s \to \mathbf{v}_s \in \mathbb{R}^d$.

Following Mikolov et al. [153], we learn two representations for each word: $\mathbf{c}_s$ represents word s when it appears as a *source*, and $\mathbf{v}_t$ for word t appearing as a *target*. Relationships act by altering $\mathbf{c}_s$ through their action on the vector space ($\mathbf{c}_s \mapsto G_R \mathbf{c}_s$). By allowing G_R to be an arbitrary affine transformation (see Section 2.3.2), we combine the bilinear form of Socher et al. [197] with translation operators of Bordes et al. [25].

We can then choose an energy function like

$$\mathcal{E}(S, R, T|\Theta) = -\mathbf{v}_T \cdot G_R \mathbf{c}_S \tag{2.25}$$

which uses parameters $\Theta = \{\mathbf{c}_i, G_r, \mathbf{v}_j, \}_{i,j \in \text{vocabulary}}^{r \in \text{relationships}}$. For this choice of energy function, we observe that the $|R| = 1$, $G_R = \mathbb{I}$ case recovers the original softmax objective described in Mikolov et al. [153], so the idealised `word2vec` model is a special case of this model. However, this energy function is problematic, as it can be trivially minimised by making the norms of all vectors tend to infinity.

While the partition function provides a global regulariser, we found that it

[1] By construction - a promising recent line of research has investigated the use of hyperbolic space to embed entities, which is particularly suited for representing tree-like structures [166].

is not sufficient to avoid norm growth during training. We therefore use as our energy function the negative cosine similarity, which does not suffer this weakness;[2]

$$\mathcal{E}(S, R, T|\Theta) = -\frac{\mathbf{v}_T \cdot G_R \mathbf{c}_S}{\|\mathbf{v}_T\| \|G_R \mathbf{c}_S\|} \tag{2.26}$$

This is also a natural choice, as cosine similarity is the standard method for evaluating word vector similarities.

Energy minimisation therefore amounts to finding an embedding in which the *angle* between related entities is minimised in an appropriately transformed relational space. It would be simple to define a more complex energy function (using perhaps splines) by choosing a different functional representation for R, but we focus in this work on the affine case.

Partition function

The partition function is the price we pay for using a joint - rather than conditional - model. The partition function Z is the normalisation constant of the probability distribution, hence

$$Z = \sum_{\omega} e^{-\mathcal{E}(\omega)} \tag{2.27}$$

where the sum is over all possible states ω.

In our case, $Z(\Theta) = \sum_{s,r,t} e^{-\mathcal{E}(s,r,t|\Theta)}$, where $\mathcal{E}$ is defined above.

[2]We also considered an alternate, more symmetric energy function using the Frobenius norm of G; $\mathcal{E}(S, R, T|\Theta) = -\frac{\mathbf{v}_T \cdot G_R \mathbf{c}_S}{\|\mathbf{v}_T\| \|G_R\|_F \|\mathbf{c}_S\|}$. However, we found no clear empirical advantage to this choice.

Inference: Persistent contrastive divergence

We estimate our parameters Θ from data using stochastic maximum likelihood on the joint probability distribution.

The maximum likelihood estimator is:

$$\Theta^* = \text{argmax } P(\mathcal{D}|\Theta) = \text{argmax } \prod_n^N P((S,R,T)_n|\Theta) \tag{2.28}$$

Considering the log-likelihood at a single training example (S, R, T) and taking the derivative with respect to parameters, we obtain:

$$\begin{aligned}\frac{\partial \log P(S,R,T|\Theta)}{\partial \Theta_i} =& \frac{\partial}{\partial \Theta_i}\left[-\mathcal{E}(S,R,T|\Theta)\right] \\ &- \frac{\partial}{\partial \Theta_i}\left[\log \sum_{s,r,t} e^{-\mathcal{E}(S,R,T|\Theta)}\right]\end{aligned} \tag{2.29}$$

Given a smooth energy function the first term is easily obtained, but the second term is problematic. This term, derived from the partition function $Z(\Theta)$, is intractable to evaluate in practice owing to its double sum over the size of the vocabulary (potentially $\mathcal{O}(10^5)$). In order to circumvent this intractability we resort to techniques used to train Restricted Boltzmann Machines and use stochastic maximum likelihood, also known as persistent contrastive divergence (PCD) [214]. In contrastive divergence, the gradient of the partition function is estimated using samples drawn from the model distribution seeded at the current training example [93]. However, many rounds of sampling may be required to obtain good samples. PCD retains a persistent Markov chain of model samples across gradient evaluations, assuming that the underlying distribution changes slowly enough to allow the Markov chain chain to mix. We use Gibbs sampling by iteratively using the conditional distributions of all variables (S, R, and T, see below) to obtain model samples.

In particular, we draw S, R and T from the conditional probability distributions:

$$\begin{aligned} P(S|r,t;\Theta) &= \frac{e^{-\mathcal{E}(S,r,t|\Theta)}}{\sum_{s'} e^{-\mathcal{E}(s',r,t|\Theta)}} \\ P(R|s,t;\Theta) &= \frac{e^{-\mathcal{E}(s,R,t|\Theta)}}{\sum_{r'} e^{-\mathcal{E}(s,r',t|\Theta)}} \\ P(T|s,r;\Theta) &= \frac{e^{-\mathcal{E}(s,r,T|\Theta)}}{\sum_{t'} e^{-\mathcal{E}(s,r,t'|\Theta)}} \end{aligned} \tag{2.30}$$

Thereby, we can estimate the gradient of $Z(\Theta)$ at the cost of these evaluations, which are linear in the size of the vocabulary, as the (generally intractable) partition function is not required.

Using this, following the objective from (2.29) further simplifies to a contrastive objective given a batch of B data samples and M model samples (each model sample obtained from an independent, persistent Markov chain):

$$\begin{aligned} \frac{\partial P(\mathcal{D}|\Theta)}{\partial \Theta_i} \simeq \frac{1}{M} \sum_{m=1}^{M} \left[\frac{\partial \mathcal{E}((S,R,T)_m|\Theta)}{\partial \Theta_i} \right] \\ - \frac{1}{B} \sum_{b=1}^{B} \left[\frac{\partial \mathcal{E}((S,R,T)_b|\Theta)}{\partial \Theta_i} \right] \end{aligned} \tag{2.31}$$

2.4 Experiments and analyses

We explore the model in several settings, using structured and unstructured data from generic, and then medical English. The data is described in Section 2.4.1.

- First, we use the model to perform triplet classification by thresholding $P(S,R,T)$ or by training a classifier on the *embeddings* of S, R, T. We use this to demonstrate how adding unstructured data can improve performance on the triplet classification task.

- Next, we examine the quality of the learned embeddings by concatenating S and T representations to predict the correct R, and show how the model can be used in an unsupervised way to infer latent relationships.
- Finally, we show how the model can be used to extend a medical knowledge base using relationships inferred from a large medical text corpus.

These experiments exploit the probabilistic nature of this model and demonstrate how knowledge graphs can be used to enhance embedding-learning using (potentially scarce) free-text corpora.

In several analyses, we consider the dual settings where either *structured* data is scarce (corresponding to a limited knowledge graph), or *unstructured* data is scarce (corresponding to limited free-text corpora). These settings are similar to multitask and transfer learning (for instance, [32, 56, 234]), which is touched upon again in later chapters.

2.4.1 Data

We distinguish between *structured* data, as from a knowledge graph, and *unstructured*, or 'free-text' data, derived from a corpus of text.

Generic English: WordNet and Wikipedia

As structured data, we use the WordNet dataset described by Socher et al. [197][3]. It contains 38,588 words and 11 types of relationships, Training data consists

[3]The dataset is available at `http://stanford.io/1IENOYH`

of true triples such as (`feeling`, `has instance`, `pride`). There are 112,581 training examples, 5,218 validation, and 21,088 test triples. Test and validation examples are labelled as true or false. False examples are built from true triplets, which are randomly selected to be in one of the sets. For each true example in the test set, a false example is created by randomly switching one of its entities. Care is taken to ensure that the new, corrupted entity can appear in that position in that relationship, therefore it must appear there with some *other* entity. This is similar to how the neural tensor model [197] (Section 2.2.2) is constructed - indeed, this dataset is developed in the same paper, and further processing details are provided there.

We derive an additional version of this dataset by stripping sense IDs from the words, which reduced the vocabulary to 33,330 words. We note that this procedure likely makes prediction on this data more difficult, as every word receives only one representation. We do this in order to produce an aligned vocabulary with our *unstructured* data source, taken to be English Wikipedia (`https://dumps.wikimedia.org/`, August 2014). We extract text using WikiExtractor[4]. We greedily identify WordNet 2-grams in the Wikipedia text. Two words are considered in a `sentence context` if they appear within a five word window. Only pairs for which both words appeared in the WordNet vocabulary are included. We draw from a pool of 112,581 training triples in WordNet with 11 relationships, and 8,206,304 triples from Wikipedia (heavily sub-sampled, see experiments). To check that our choice to strip sense IDs was valid, we also created a version of the Wikipedia dataset where each word was tagged with its most common sense from the WordNet training corpus. We found that this did not significantly impact our results, so we chose to continue

[4] `http://bit.ly/1Imz1WJ`

with the sense-stripped version, preferring to collapse some WordNet identities over assigning possibly-incorrect senses to words in Wikipedia.

We use this 'generic' English data for experiments in Sections 2.4.3 and 2.4.4.

Medical English: SemMedDB and MSKCC Text Notes

This is the data used in experiments described in Section 2.4.5.

In the medical setting, the unstructured data is a corpus of de-identified clinical notes written by physicians at Memorial Sloan Kettering Cancer Center (MSKCC). We process raw text by replacing numbers with generic tokens such as `HEIGHT` or `YEAR`, and removing most punctuation. In total, the corpus contains 99,334,543 sentences, of which 46,242,167 are unique. This demonstrates the prevalence of terse language and sentence fragments in clinical text; for example the fragment `no known drug allergies` appears 192,334 times as a sentence.

The tokens of interest are chiefly Concept Unique Identifiers (CUIs) from the Unified Medical Langauage System (UMLS) [20]. These represent discrete *medical concepts*, which may require several words to describe, for example: `C0023473: chronic myelogenous leukemia`. We consider it more meaningful and interesting to consider relationships between CUIs rather than words themselves, when possible.

We identify CUIs in the text by greedily matching against strings associated with CUIs (each CUI can have multiple such strings). This results in 45,402 unique CUIs, leaving 270,100 non-CUI word tokens. We note that the

MetaMap [7] tool is a more sophisticated approach for this task, but found it too inefficient to use on a dataset of our size. To generate (S, R, T) triples, we consider two words in a `appears in a sentence with` relationship if they are within a five-word window of each other.

We exploit the existence of `SemMedDB` [118], a database of semantic predications in the form of subject-relationship-object triples, where the subjects and objects are CUIs. These were derived from PubMed abstracts using the tool SemRep [185], and contain triplets such as `C0027530(Neck)` is `LOCATION OF` `C0039979(Thoracic Duct)` or `C0013798(Electrocardiogram) DIAGNOSES` `C0026269(Mitral Valve Stenosis).` `SemMedDB` contains 82,239,653 such statements, of which 16,305,000 are unique. This covers 237,269 unique CUIs.

Since the distribution of CUI/token frequencies has a long tail in both data sources, we threshold tokens by their frequency. Firstly, tokens (words of CUIs) must appear at least 100 times in either dataset, and then at least 50 times in the pruned datasets. That is, in the first round we remove sentences (in `EHR`) or statements (in `SemMedDB`) containing 'rare' tokens. In addition, the 58 relationships in `SemMedDB` also exhibit a long-tailed frequency distribution, so we retain only the top twenty.

From this pool of (S, R, T) triples (from `EHR` and `SemMedDB`) we create fixed test sets (see next subsection) and smaller datasets with varying relative abundances of each data type, using 0, 10, 50, 100, 500, and 1000 thousand training examples. The final list of tokens has size $W = 45,586$, with 21 relationships: twenty from `SemMedDB` and an additional `appears in sentence with` from `EHR`. Of the W tokens, 7,510 appear in both data sources. These overlapping tokens are critical to ensure embeddings derived from the knowledge

graph are consistent with those derived from the free text.

2.4.2 Experimental details and implementation

We use `Adam` [120] to adapt learning rates and improve numerical stability. We used the recommended hyperparameters from this paper; $\lambda = 1 - 10^{-8}$, $\epsilon = 1 - 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Unless otherwise stated, hyperparameters specific to our model were: dimension $d = 100$, batch size of $B = 100$, learning rate for all parameter types of $\alpha = 0.001$, and three rounds of Gibbs sampling to obtain model samples. We use a l_2 regulariser with strength 0.01 on G_r parameters (relationship weight matrices).

2.4.3 Triplet classification

Classification on WordNet

We use our model to solve the basic prediction task described in Socher et al. [197]. In this case, the model must differentiate true and false triples, where false triples are obtained by corrupting the T entry in the triple, e.g. $(S, R, T) \rightarrow (S, R, \tilde{T})$ (where $(S, R, \tilde{T})$ doesn't appear in the training data). The corrupted version of T must exist in relationship R with some other S in the dataset, to make the problem more challenging. The 'truth' of a triple is evaluated by its energy $\mathcal{E}(S, R, T)$, with a relationship-specific cut-off chosen by maximizing accuracy on a validation set. By learning explicit representations of each of the 38,588 entities in WordNet, our approach most closely follows the *'Entity Vector'* task in Socher *et al.* This is to be contrasted with the *'Word Vector'* task,

where a representation is learned for each word, and entity representations are obtained by averaging their word vectors. We elected not to perform this task because we are not confident that composition into phrases through averaging is well-justified. Using the validation set to select an early stopping point at 66 epochs, we obtain a test set accuracy of 78.2% with an AUROC of 85.6%.

The 'Neural Tensor Model' (NTN) described in Socher et al. [197] and Section 2.2.2 achieves an accuracy of around 70% on this task, although we note that the simpler Bilinear model (also described in Socher et al. [197]) achieves 74% and is closer to the energy function we employ. The improved performance exhibited by this simpler Bilinear model was also noted by [239].

Other baselines reported by Socher *et al.* were a single layer model without an interaction term, a Hadamard model [24] and the model of Bordes et al. [23] which learns separate left and right relationship operators for each element of the triple. These were outperformed by the Bilinear and NTN models, see Figure 4 in Socher et al. [197] for further details. Hence, our model outperforms the two previous methods by more than 4%.

As a preliminary test of our model, we also considered the `FreeBase` task described by Socher et al. [197]. Initial testing yielded an accuracy of 85.7%, which is comparable to the result of their best-performing model (NTN) of about 87%. We chose not to further explore this dataset however, because its entities are mostly proper nouns and thus seemed unlikely to benefit from additional semantic data.

Adding unstructured data for triplet classification

In this case, we assume structured data is scarce, and analyse the effect of augmenting with unstructured data. That is, we take a small set of X examples from WordNet (our structured data source), where X is between 1000 and 10000 (x-axis in Figure 2.1, right), and augment it with 10,000 examples from Wikipedia (the unstructured data source).

We evaluate the model using the triplet-classification task used in earlier analyses - note that the test set only contains triplets from WordNet. Unstructured data provides triplets of the form (S, R_u, T) where R_u is the single relationship 'appears in a sentence with'. This relationship does not exist in WordNet, so we would not expect the Wikipedia data to explicitly provide information about any relationships appearing in WordNet. An improvement in test-set performance must therefore be related to the *general semantic* information contained in the Wikipedia examples. As an extreme example, if it becomes apparent due to the Wikipedia data that T_1 and T_2 are synonyms, then if the model has observed $P(S, R, T_1)$ in WordNet, it can easily classify $P(S, R, T_2)$, even if T_2 never appeared in the WordNet data. This transfer is possible because synonymy, which is captured easily by distributional semantics, implies other relationships.

In practice, we found it necessary to balance the amount of data from WordNet and Wikipedia. We do this by using a scaling factor κ on examples from Wikipedia, effectively scaling down gradients from this data source. **Figure 2.1** shows accuracy on this task as κ and the amount of WordNet data vary. To find the improvement associated with *unstructured data*, we compared accuracy at $\kappa = 0$ with $\kappa = \kappa^*$ (where κ^* gave the highest accuracy on the validation set; marked with $*$ in the figure) - this is show in the right panel. We find that

including free text data quite consistently improves the classification accuracy, particularly when structured data is scarce.

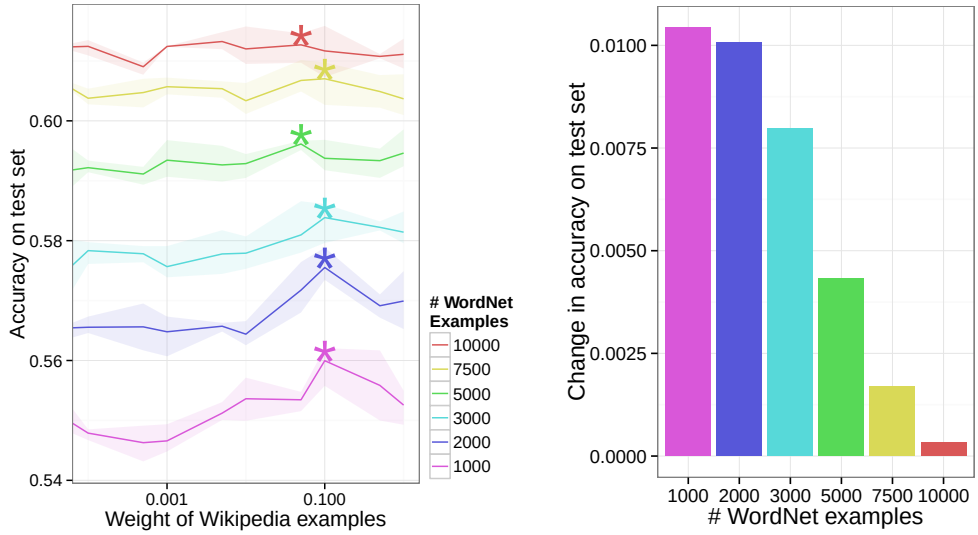


Figure 2.1: **Unstructured data improves triplet classification**. In addition to training on a set of known relationships, we use unstructured data from Wikipedia with varying weight (x-axis) during training. As before, with the goal to predict if a triple (S, R, T) is true by using its energy as a score. A validation set is used to determine the threshold below which a triple is considered 'true'. The solid line denotes the average of three independent experimental runs; shaded areas show the range of results. The bar plot on the right shows the difference in accuracy between $\kappa = 0$ and $\kappa = \kappa^*$, where κ^* gave the highest accuracy on a validation set. Significance at 5% (paired t-test) is marked by an asterisk. We find then that unstructured Wikipedia can improve relationship learning in cases when labelled relationship data is scarce.

2.4.4 Embedding quality

In these analyses, the focus is no longer on the ability to classify triplets as true or false using the energy function, but rather on the quality of the word representations themselves. Evaluating 'unsupervised' representations is a topic of ongoing research, and consensus has not yet been reached on the optimal strategy (if it is unique).

Here, we perform evaluation as follows: We take embeddings of words S, T and use them as inputs to a supervised multi-class classifier. The task for a given (S, R, T) triple is to predict R given the vector formed by concatenating $\mathbf{c}_S$ and $\mathbf{v}_T$ (the representations of word S as source and T as target). Note that this does not use G_R - the embeddings are *untransformed* and exist in the original space. This is intended to mimic a possible use-case for such embeddings, since the relationship in question (and therefore the transformation G_R to the new space) may not be known *a priori*. We generate this dataset by taking (S, R, T) triplets, and using R as the label to be predicted. However, since we already know that S and T are related through R, we can re-use our dataset to invent this downstream task for the data.

Augmenting Wikipedia embeddings with WordNet

In this case, we assume *unstructured text data* from Wikipedia is restricted, and vary the quantity of structured data from WordNet. While the analysis in Section 2.4.3 is especially interesting for the case of generic English, where unstructured data is highly available (as is the case for Wikipedia), this setting is more interesting for medical purposes, where text corpora are smaller and more restricted, but knowledge graphs are available. We explore the medical setting in the context of knowledge graph completion in Section 2.4.5.

To avoid testing on the training data (since the embeddings are obtained using the WordNet training set), we perform this procedure once for each relationship (9 times - excluding `appears in sentence with`), each time removing from the training data *all* triples containing that relationship. Specifically, we consider the WordNet relationships `has instance`,

`domain region`, `subordinate instance of`, `member holonym`[5], `has part`, `part of`, `member meronym`[6], `synset domain topic`, `type of`. Relationships `similar to` and `domain topic` were omitted for technical reasons. This means that any information about relationship R contained in the embeddings must have been learned implicitly through the other relationships. In the case where relationships are the inverse of each other (e.g. holonymy/meronymy) it is more obvious that information can be transferred, but not all relationships in this set have inverses.

We use a random forest classifier trained on the WordNet validation set using five-fold cross-validation.

Figure 2.2 shows the $F1$ score of the multi-class classifier on the left-out relationship for different combinations of data set sizes. We also trained `word2vec` [153] on a much larger Wikipedia-only dataset (4,145,372 *sentences*) and trained a classifier on its vectors; results are shown as black lines.

Since we are interested in how adding structured (WordNet) data improves, the most interesting results are the two right-most bars in Figure 2.2 - these correspond to a fixed set of 50,000 examples from Wikipedia with either 10,000 or 50,000 examples from Wordnet respectively. We see that in 6 out of 9 relationships, the addition of WordNet data improved performance in this task, and using any WordNet information improves over the `word2vec` baseline (black lines). This indicates that even data about *unrelated* relationships provides information to produce vectors that are semantically richer overall. These results illustrate that embeddings learned from limited free text data can be improved by additional, unrelated relationship data.

[5]Concept S is a holonym of T if Ts are parts of S, e.g. 'dog' is a holonym of 'paw'.
[6]Meronymy is the opposite of holonymy - 'paw' is a meronym of 'dog'.

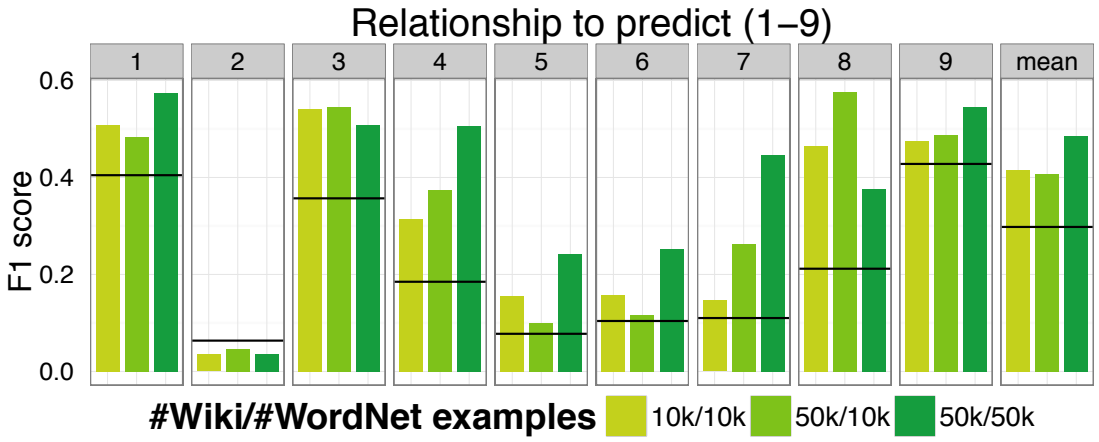


Figure 2.2: **Relationship data improves learned embeddings.** We apply our algorithm on a scarce set of Wikipedia co-occurrences (10k and 50k instances) with varying amounts of additional, unrelated relationship data (10k and 50k relations from WordNet). We test the quality of the embedding by measuring the accuracy on a task related to nine relationships (see text for which). In each case, we used eight of the relationships together with the Wikipedia data to learn representations that are then used in a subsequent supervised learning task to predict the remaining ninth relationship based on the representations using random forests. Black lines denote results from `word2vec` trained on a Wikipedia-only with 4,145,372 *sentences*.

Unsupervised learning

In this experiment, we explore the ability of the model to learn embeddings from co-occurrence data *alone*, without specifying the relationships it should use. This is possible because the model described in Section 2.3 can gracefully deal with missing elements in observed triplets (for instance missing observed relationships).

We can consider the partially observed triple as a superposition of all possible completions of that triple, each weighted by its conditional probability given the observed elements, using (2.30). In the fully-observed case, we assume all triplets (S, R, T) appear in the data with probability 1 (that is, we take a Bayesian approach and assume the data is given). In the semi-observed case,

we use the model to add virtual observations (S, r, T) to the dataset, but weight these with their probability given what is observed, that is $P(r|S, T)$. That is, if we observe $(S, , T)$, we imagine that we could have observed (S, r, T) with probability $P(r|S, T)$, for all relationships r. This has the effect of producing a gradient which is a weighted sum over these possible observations (we assume the probability is a constant for the purpose of computing the gradient). Hence,

$$\frac{\partial P(S, , T; \Theta)}{\partial \Theta} = \sum_{r}^{R} P(r|S, T) \frac{\partial P(S, r, T; \Theta)}{\partial \Theta} \tag{2.32}$$

In the fully-observed case, this weighting is simply a delta function on the observed state.

When using the model with just one relationship (trivially the identity), the model effectively reverts to `word2vec`. However, if we add a 'budget' of relationships (in our experiments we use 1, 3, 5, 7, 11), the model can potentially achieve lower energy solutions by selectively applying these relationship transformations, allowing some concept pairs to be closer in a context-dependent way, where the set of contexts is not specified *a priori*.

The intuition is that we want to test whether textual context alone has substructure that we can capture with latent variables.

We generate a training set of one million word co-occurrences from Wikipedia (using a window size of 5 and restricting to words appearing in the WordNet dataset, as described earlier), and train different models for each number of latent relationships. Inspired by earlier experiments testing the utility of supplanting WordNet training data with Wikipedia examples, we decide to test the ability of a model purely trained on Wikipedia to learn word and relationship representations which are predictive of WordNet triplets, *without* having

seen any data from WordNet. We use the setup as in the previous analysis - we concatenate $\mathbf{c}_S$ and $\mathbf{v}_T$ and feed it into a random forest to predict the relationship R between S and T. The relationships are taken from WordNet, and need not correspond to the latent relationships learned by the model in this case.

As a baseline we start with $|R| = 1$ to test how well word embeddings from context alone can perform, indicated by the leftmost bar in **Figure 2.3**. Once again, we use `word2vec`-trained embeddings as a baseline, shown in black (this is the same data as depicted in Figure 2.2.) We then proceed to train models with more latent relationships. We observe that, especially for some relationship prediction tasks, including this flexibility in the model produces a noticeable increase in $F1$ score on this task. Since we evaluate the embeddings alone, and not for example likelihood, this effect cannot be explained by the additional parameters introduced by the latent relationships, and must be due to a shift in the content of the vectors.

A possible explanation for this phenomenon is that the model discovers contextual subclasses which are indicative of WordNet-type relationships.

We note that we did not perform an exhaustive search of the hyperparameter space; better settings may yet exist. Nonetheless, although the absolute improvement in $F1$ score yielded by this method is modest, we are encouraged by the model's ability to exploit latent variables in this way.

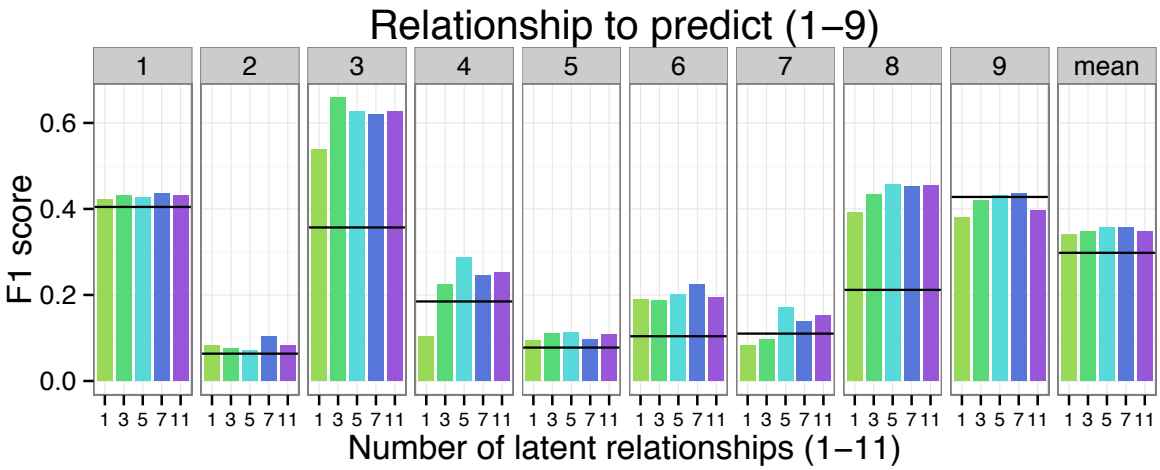


Figure 2.3: **Unsupervised learning of latent relationships improves embeddings.** We train a fully unsupervised algorithm with 1, 3, 5, 7 and 11 possible latent relationships on one million Wikipedia sentences. Initialisation is at random. To test the quality of the resulting *embeddings*, we use supervised learning of nine WordNet relationships with random forests. Depending on the relationship at hand, the use of multiple latent relationships during training leads to slightly, but consistently better accuracies using the computed embeddings for every of the nine relationships and also on average. Hence, the resulting embeddings using latent relationships can be said to be of higher quality. Once again, black lines show results using `word2vec`.

2.4.5 Extending a medical knowledge graph

Having examined developed and analysed the approach on generic English in the previous sections, in these analyses we now focus on medical English. Specifically, we use as unstructured data de-identified text notes written by clinicians at MSKCC, where terms from the UMLS ontology [20] have been identified. The medical ontology we augment it with is SemMedDB, which contains predicates between UMLS concepts extracted from PubMed citations [118]. The data is described in detail in Section 2.4.1.

Augmenting EHR embeddings with SemMedDB

Experimental design As the model defines conditional distributions for each element of a triple given the remaining two (Equation 2.30), we can test the ability to predict new components of a knowledge graph. For example, by selecting the best R given S and T, we predict the relationship (the type of edge) between tokens S and T using $P(R|S,T)$.

Without loss of generality, we describe the procedure for generating the test set for the R task. We select a random set of S,T pairs appearing in the data. For each pair, we record all entities r which appear in a triple with them, removing these triples from the training set. The $S,T \rightarrow \{r_i\}_i$ task is then recorded in the test set. Evidently, there may be many correct completions of a triple; in this case we expect the model to distribute probability mass across all answers. How best to evaluate this is task-dependent; we consider both the *rank* and the *combined probability mass* in these experiments.

Results Figure 2.4 shows results for the task of predicting R given S and T. The model produces a ranking of all possible Rs (high probability $\rightarrow$ low rank) and we report the mean reciprocal rank of the *lowest-ranked* correct answer over the test set. We use this metric to evaluate the utility of these predictions in *prioritising* hypotheses to test: we would like *any* correct answer to be ranked highly, and don't apply a penalty for a failure to capture alternative answers. Results for our model are marked by `bf`[7] and `bf++`. The latter model uses an additional 100,000 training examples from the `EHR`: these are 'off-task' information. As a baseline we consider a random forest trained to predict R given the

[7] `bf` stands for 'brí-focal', which means *word meaning* in Irish.

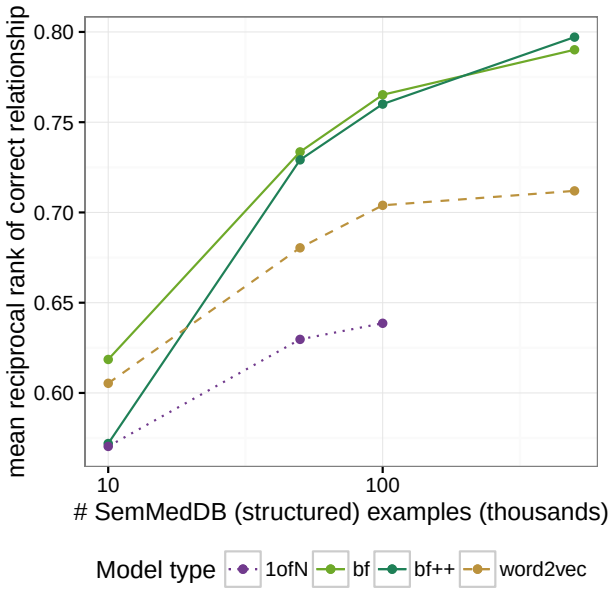


Figure 2.4: **Predicting the relationship between concepts.** With more structured data, the model can better predict the correct relationship for a given (S, T) pair. `bf++` has an additional 100,000 triples from `EHR`: with little structured data, so much off-task information is harmful, but provides some benefit when there is enough signal from the knowledge graph. Baselines are a random forest taking $[f(S) : f(T)]$ as an input to predict the label R, where the feature representation f is either a 1-hot encoding (`1ofN`) or 200-dimensional `word2vec` vectors trained on PubMed. `1ofN` proved too computationally expensive for large data.

concatenation $[f(S) : f(T)]$, where the representation f is either: a) `1ofN`: each token has a binary vector of length W (W = 45,586), b) `word2vec`: each token has a 200-dimensional vector obtained by running `word2vec` [152] trained on PubMed [160]. We note that the PubMed corpus contains over 2 billions tokens, far more data than was available to `bf`. We additionally trained `TransE` [25] on this data, but it proved unsuited to the task (data not shown).

As we can see, adding examples from `SemMedDB` improves performance for all model types, but `bf` seems to make better use of the additional data. In spite of its very large input vector size ($2W = 91172$), `1ofN` struggles, likely as it treats all tokens as independent entities. We note that for `bf++`, performance is

degraded when the amount of structured data is low. This is consistent with our earlier observations on non-medical data, as the quantity of 'off-task' information added is in this case comparable to that of 'on-task'. Interestingly however, the model appears slightly *better able* to exploit more structured data when some 'semantic background' is provided by `EHR`.

Experimental design As mentioned, the model is capable of combining structured and unstructured data. In earlier experiments we observed that classification performance on a knowledge base could be improved by addition of unstructured data. However, the task in that case was quite 'easy'; the model simply needed to differentiate between true and false triples. Here we consider the harder problem of correctly selecting which entity would complete the triple.

In addition to possibly improving performance, access to unstructured data provides the opportunity to *augment* the knowledge base. That is, we can predict relationships for tokens *not appearing* in `SemMedDB`. This uses the joint embedding of all tokens into one vector space, regardless of their data source. The geometric action of the relationships learned from `SemMedDB` can then be applied to the representation of any token, such as those uniquely found in `EHR`. We note that this procedure amounts to *label transfer* from structured to unstructured examples, which can be understood as a form of semi-supervised learning.

To generate ground truth for this task, we select some tokens $\{T_i\}$ (these could appear as S or T entities) found in both `SemMedDB` and `EHR` and remove them from `SemMedDB`, recording them to use in the test set. Put another way, as

in the previous setting, during the 'random' selection of S, T (still wlog) pairs, we make sure all of these recording them to use in the test set. Put another way, as in the previous setting, during the 'random' selection of S, T (still wlog) pairs, we make sure all T_i in the deletion list are included, alongside any other tokens which appear in a `SemMedDB`-derived relationship with them. The task is then to use purely *semantic* similarity gleaned from `EHR` to place these tokens in the embedding space such that the action of relationship operators is still meaningful.

Results Figure 2.5 shows results on all three tasks (predicting S, R, T given the remaining two), as a function of the type of test example. The right column of results is for test entities involving at least one element *not appearing* in `SemMedDB`. As we are now interested in the embeddings themselves we report the probability mass of true entities, feeling this better captures the information contained in the embeddings. That is, it is no longer sufficient for the model to correctly predict *a single* answer, we want it to assign appropriate probability mass to *all* correct answers. The dotted grey lines demonstrate the random baseline, where all tokens are equally likely. The probability mass assigned by the baseline is therefore equal to k/W (or k/R) where k is the average number of correct options in that task type.

There are several observations to be made here:

- Most of the time, performance is best with a non-zero, but *relatively small* amount of `EHR` data (x-axis). This supports our observations that off-task information improves embeddings, but can 'drown out' signal if it dominates relative to the on-task examples. This can be improved by including

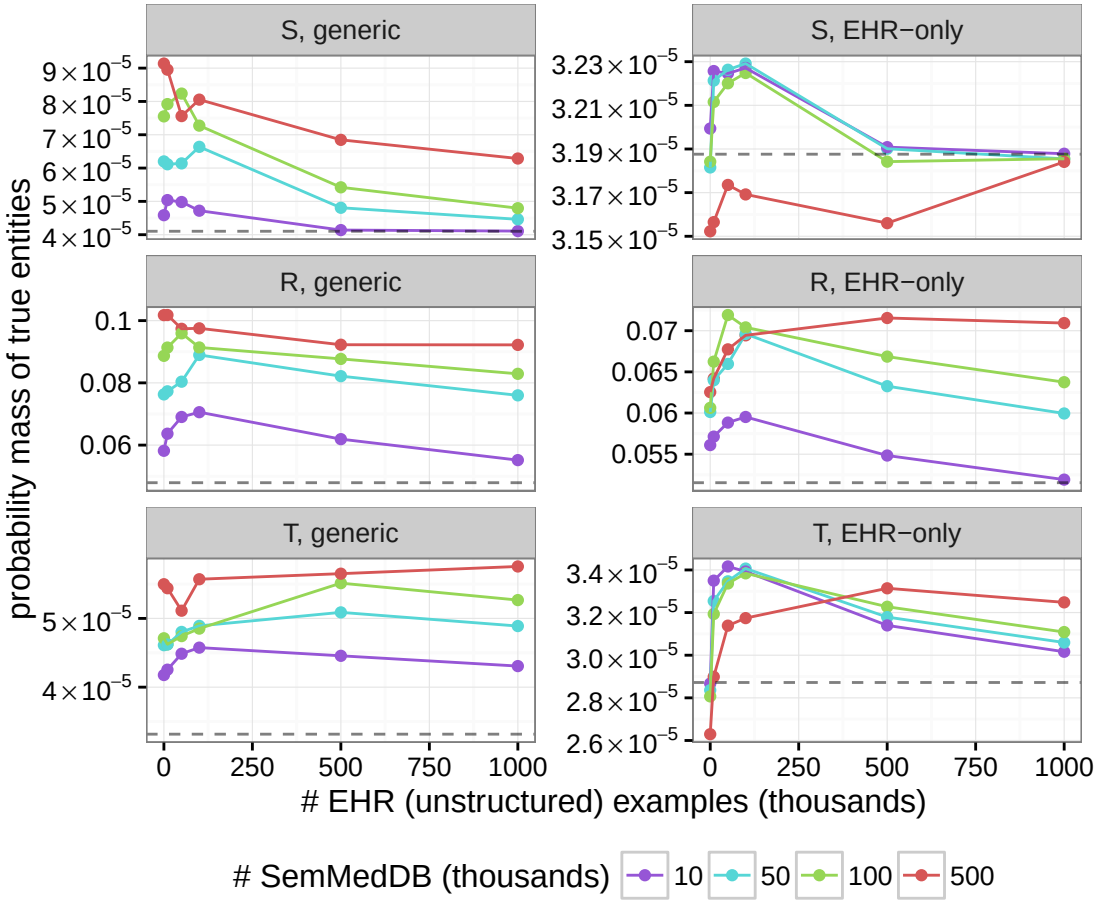


Figure 2.5: **Probability mass assigned to correct answers in the knowledge graph completion and knowledge transfer task.** The right column shows results for test triples where at least one of S and T is found *only* in `EHR`, and therefore represents the *knowledge transfer* setting. Information about relationships found in `SemMedDB` must be transferred through the joint embedding to enable these predictions. Grey dotted lines represent a random-guessing baseline.

a pre-factor on gradient contributions from the off-task data to adjust their contribution relative to the structured examples, as demonstrated in earler sections.

- The `EHR-only` setting is much harder, as anticipated. In the case of S and T it is comparable to the random baseline. For R however, the model successfully assigns probability mass when there is enough `SemMedDB` data

available.

- The S and T tasks are not symmetric. The S task features slightly more correct options on average than T (1.87 and 1.5 respectively, for the `generic` task), but this does not account for the difference in proportional performance relative baseline, especially at low `EHR` abundance. A possible explanation is the energy function (Equation 2.26): it does not treat S-type and T-type variables identically. However, experiments using the Frobenius norm of G_R in the denominator of $\mathcal{E}$ did not remove asymmetry, so it is likely that the tasks are simply not equivalent. This could arise due to bias in the directionality of edges in the knowledge graph.

We conclude that it is possible to use the joint embedding procedure to predict R for pairs of S, T entities even if they do not appear in `SemMedDB`. For the harder S and T tasks, the model generally succeeds in improving visibly over baseline, but its assignments are still quite 'soft'. This may reflect premature stopping during training (most results reported were before 50 epochs had elapsed), an insufficiently powerful model formulation, or an excess of noise in the training data. Many predicates in `SemMedDB` are vague, and some relationships lend themselves to a one-to-many situation, for example `part of`, or `location of`. A core assumption in our model is that a token with fixed vector representation can be transformed by a single affine transformation to be similar to its partner in a relationship. Many-to-one (or vice-versa) type relationships requires that multiple unique locations must be mapped to the same point, which necessitates a rank-deficient linear operator or a more complex transformation function (one which is locally-sensitive, for example). Future work in relational modelling must carefully address the issue of many-to-many and hierarchical relationships.

2.5 Conclusions and future work

This chapter has described a probabilistic generative model combining structured and unstructured data, for the purpose of learning representations of words and concepts and the relationships between them. These relationships are represented as affine transformations on the vector space in which the words are embedded. This model can be used for classifying facts and extending the structured data source through knowledge graph completion. Moreover, the model is able to use *latent* relationships to enhance its representation-learning capacity.

Several avenues for further research are available, for example:

- This chapter has focused on learning representations of words or clinical concepts. Representing higher-order language elements, such as phrases, sentences or even paragraphs, is a topic of ongoing research. The specifics of medical English raise questions as to how best to approach this composition problem. For example, sentence order within a clinical text note is contextually important if a doctor is listing possible diagnoses.
- This chapter is motivated by the need to develop domain-specific text representations. To this end, the potential to transfer embeddings learned from large, semi-related corpora has not been explored here. This practice of adapting embeddings for specialised use-cases has only been studied to a limited extent in the medical domain [243], and could be developed further.
- Relationships between concepts are represented here as affine transformations. While this was chosen to capture existing methods for representing

relationships, it may nonetheless be overly restrictive in certain cases. For example, in the case of hierarchical relationships, which are typically one-to-many, the model is forced to map many points to a single location, necessitating a singular transformation matrix (in theory), or in the reverse case, a one-to-many mapping, which is not possible for a deterministic transformation. Studying the types of relationships and the most appropriate way to represent them, is a question for further research.

- Ultimately the purpose of representation-learning is to use those representations. There are many interesting applications of word-level representations beyond knowledge graph completion: to reduce variation in language use between clinicians by identifying and collapsing similar terms, to enable partial matches while searching for clinical trials (or other information retrieval tasks), or to characterise the similarity between types of clinical variables used in time-series applications. For example, the early warning system described in Chapter 5 uses variables which are treated as mutually independent, despite having varying levels of semantic similarity - for example, systolic and diastolic blood pressure are more similar than systolic blood pressure with creatinine. Having prior knowledge about the similarity between medical *concepts* enables measurements of these concepts to be related to each other in a more nuanced way.

CHAPTER 3

LONG-MEMORY RECURRENT NEURAL NETWORKS

> All moments, past, present and future, always have existed, always will exist.
>
> Kurt Vonnegut, *Slaughterhouse-Five*

Individual contributions *The work in this chapter was a collaboration between me and my supervisor Gunnar Rätsch, who provided supervision and guidance. Models, experiments and other details were implemented and executed by me. This work was published at the 31st AAAI Conference in Artificial Intelligence in 2017 [100].*

Recurrent neural networks (RNNs) are a class of artificial neural network with a recurrent structure (see Figure 3.1):

In the simplest version of the RNN, the internal and external connections take the following form:

$$\begin{aligned} h_t &= \sigma\left(U h_{t-1} + V \mathbf{x}_t + \mathbf{b}\right) \\ o_t &= W h_t + b_o \end{aligned} \tag{3.1}$$

where σ is some nonlinear function, such as $\tanh$. As data ($\mathbf{x}_t$) appears sequentially (indexed by t), the hidden state h_t is repeatedly updated according to its recurrence relation.

This architecture gives the RNN two special properties:

1. The total length (T) of input sequences do not need to be specified or

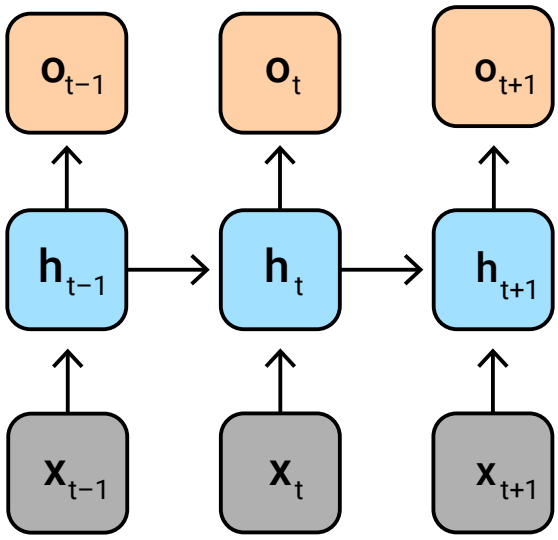


Figure 3.1: **Schematic of a recurrent neural network.** The hidden state h_t of the network at time t depends on the previous hidden state (h_{t-1}) and the observed data ($\mathbf{x}_t$). The model can produce outputs o_t, such as predictions, at each time step.

known ahead of time. In principle, sequences of arbitrary length can be processed. In practice, this is not the case, as we see in the next section.

2. The hidden state h_t can be considered a representation of the data seen up to time t. Outputs (e.g. predictions) at time t depend on all data observed up to t, by way of h_t.

These properties make RNNs well-suited to processing and modelling time series or other sequential data. Due to the recurrent structure, RNNs can be considered a type of nonlinear auto-regressive model.

In practice, given a sequence of fixed length (T), the RNN can be 'unrolled' into a network of depth T (see Figure 3.1). For this reason, RNNs can be considered to be 'very deep' networks with many shared parameters, as the connections between subsequent layers are the same, given by the weights in the recurrence relation (Equation 3.1).

Many variants of this basic architecture have been proposed, based on different limitations of the model, both theoretical and practical. For example, bidirectional RNNs [191] allow 'temporal' dependencies in the hidden state to run both directions, pixel RNNs [217] allow the hidden state to depend on its (spatial) neighbours, and the clockwork RNN [124] allows for different sampling frequencies in the dimensions of the input data.

In this chapter, we focus on the challenge of *long* time-series. While RNNs can theoretically handle input data of arbitrary length, in practice they exhibit a pathology known as 'vanishing/exploding gradients' as the input sequence grows longer. This makes modelling long data challenging. In the next sections, we describe how this problem arises, attempts which have been made to overcome it, and a new proposal exploiting Lie algebras.

3.1 Background on long-memory RNNs

The time-series generated by one's health history is long. Starting from before we are born, factors in our life can influence our health trajectory, and events can leave lasting effects. A successful model of health needs to be able to 'retain' important events in its state representation. However, as we will now show, standard RNNs are doomed to struggle on sufficiently long time-series.

3.1.1 Vanishing and exploding gradients

Through its recurrent structure, the prediction of an RNN at time t depends implicitly on data observed at all earlier timepoints.

$$\begin{aligned} o_t &= Wh_t + b_o \\ &= W\sigma\left(Uh_{t-1} + V\mathbf{x}_t + \mathbf{b}\right) + b_o \\ &= W\sigma\left(U\sigma\left(Uh_{t-2} + V\mathbf{x}_{t-1} + \mathbf{b}\right) + V\mathbf{x}_t + \mathbf{b}\right) + b_o \\ &= \cdots \end{aligned} \tag{3.2}$$

Unfortunately, although an arbitrary datapoint $x_{t-\tau}$ appears in this expression, the *influence* it has on the prediction can vary purely due to its location in the sequence. This problem was first described in Hochreiter's 1991 masters thesis [94].[1] In this work, the error signal (backpropagated from the prediction error) was analysed at intermediate steps in the network. They demonstrated that the error signal would either increase or decrease exponentially with depth, depending on properties of the weights and the nonlinearity in the recurrence relation. Therefore, this problem is known as the 'vanishing/exploding gradient problem'. In the former case, learning becomes very slow, and the model effectively 'forgets' old inputs. In the latter case, large gradients cause oscillations in the network weights, inhibiting convergence. Exploded gradients result in oscillations in the network weights, and vanished gradients result in slow learning.

We can follow Pascanu et al. [174] and Arjovsky et al. [6] and demonstrate, from Equation 3.2, how the pathology arises. Suppose the cost C is a function of the final output of the network o_T [2].

[1] The same problem was also reported by Bengio et al. [14].

[2] A technique known as target replication [137, 165] sidesteps the vanishing/exploding gradi-

It is easy to demonstrate how the dependence of the output on the input at a given time-point vanishes with the temporal distance. Consider the norm of the gradient of the cost C with respect to the data at time τ, and use submultiplicativity of the norm to write;

$$\left\| \frac{\partial C}{\partial \mathbf{x}_\tau} \right\| \leq \left\| \frac{\partial C}{\partial \mathbf{x}_T} \right\| \left(\prod_{t=\tau}^{T-1} \left\| f'\left(U\mathbf{h}_t + V\mathbf{x}_t + \mathbf{b} \right) \right\| \|U\| \right) \left\| \frac{\partial \mathbf{h}_\tau}{\partial \mathbf{x}_\tau} \right\| \tag{3.3}$$

where f' is a diagonal matrix giving the derivatives of the nonlinearity. If we have a nonlinearity such that $\|f'\| <= \kappa$, then we can simplify the expression to:

$$\left\| \frac{\partial C}{\partial \mathbf{x}_\tau} \right\| \leq \left\| \frac{\partial C}{\partial \mathbf{x}_T} \right\| \|\kappa U\|^{T-1-\tau} \left\| \frac{\partial \mathbf{h}_\tau}{\partial \mathbf{x}_\tau} \right\| \tag{3.4}$$

As a function of τ, this norm is thus proportional to $\|\kappa U\|^{T-1-\tau}$. As long as $\|\kappa U\| < 1$, or $\|U\| < \frac{1}{\kappa}$, the dependence of C on $\mathbf{x}_\tau$ vanishes as $T - \tau$ grows larger. This is also shown in Pascanu et al. [174], where they formulate it in terms of the eigenvalues of U.

The assumption that $\|f'\| <= \kappa$ holds for many popular nonlinearities, for $\kappa = 1$ (which is commonly assumed in these analyses):

$f(x)$	$f'(x)$	max $\|f'(x)\|$
$\tanh(x)$	$1 - \tanh^2(x)$	1
$\frac{1}{1+\exp(-x)}$	$f(x)(1 - f(x))$	$\frac{1}{4}$
$f(x) = \begin{cases} x & x > 0 \\ 0 & x <= 0 \end{cases}$	$f'(x) = \begin{cases} 1 & x > 0 \\ 0 & x <= 0 \end{cases}$	1

Although we see that when $\|U\| > 1$, the right-hand side in 3.4 will grow exponentially, this is not sufficient to demonstrate exploding gradients as this expression is merely an upper bound and does not guarantee pathological behaviour. Further analysis of the exploding gradient problem is available in

ent problem by requiring the network to predict the final label at every intermediate timepoint, but does not fundamentally solve the issue of learning long-term dependencies.

Hochreiter [94]. We can motivate it intuitively by noting that, in computing the gradient of the loss with respect to the transition matrix U, there will be a contribution from every time step of the sequence, which results in a term of order $\|U\|^T$ (among other terms), where T is the length of the sequence.

3.1.2 Long short-term memory

After describing the exploding/vanishing gradient problem in 1991 [94], Hochreiter and Schmidhuber [95] proposed an architecture designed to avoid it. This model, known as the long short-term memory network (LSTM)[3], exploits a gating mechanism to explicitly allow the network to selectively update its hidden state. This allows information to be retained or forgotten depending on new input.

The LSTM modifies the recurrence relation of the vanilla RNN (Equation 3.1) with the additional of some control logic.

$$\begin{aligned} i_t &= \sigma(W_i \mathbf{x}_t + U_i m_{t-1} + b_i) \\ j_t &= \sigma(W_j \mathbf{x}_t + U_j m_{t-1} + b_j) \\ f_t &= \sigma(W_f \mathbf{x}_t + U_f m_{t-1} + b_f) \\ o_t &= \sigma(W_o \mathbf{x}_t + U_o m_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ F(j_t) \\ m_t &= \sigma(o_t) \circ F(c_t) \end{aligned} \tag{3.5}$$

In this case, we see that the state c_t is updated according to the values of the forget gate f_t, which tends to retain the same c_t, or the input gate i_t, which would

[3]Short-term memory in this sense refers to the activations of the network, and long-term memory is in the weights.

rather modify update c_t with j_t, the candidate new state. The state m_t is the 'output' of the cell, gated by the output gate o_t, depending on the value of the state c_t. Various modifications to the LSTM architecture have been proposed since its inception, including the addition of peephole connections [69], and modified units such as the gated recurrent unit [41] and variants [34] which also exploit the paradigm of controlling information flow explicitly.

3.1.3 Orthogonal and unitary RNNs

The core idea behind the LSTM solution to pathological gradients is to modify the architecture to allow for better-behaved gradients due to more desirable information flow. This basic premise has also been used for non-recurrent deep networks such as residual networks [88]. Rather than modifying the architecture itself, an alternate approach to addressing poorly-behaved gradients, thus enabling the training of a long-memory RNN, is based on Equation 3.4. Since the norm of U dictates the behaviour of the norm of the gradient (assuming constant $|f'(x)|$, the idea is to constrain U such that $\|U\| = 1$.

One way to achieve this is to enforce that U is orthogonal (or unitary) - in this case, the eigenvalues of U satisfy $|\lambda| = 1$, implying $\|U\| = 1$.

The challenge is then to devise a strategy for ensuring that U is orthogonal (or unitary) throughout training. An orthogonal initialisation of U is not sufficient, since

$$U' = U - \alpha \frac{\partial C}{\partial U} \tag{3.6}$$

is not in general orthogonal for orthogonal U, so standard gradient updates will quickly break the unitary requirement.

Various kinds of solutions have been proposed:

Initialisation and soft constraints

A surprisingly effective 'solution' to the challenge is to abandon the idea, and merely initialise the transition matrix to be unitary or orthogonal. This is the approach taken in Le et al. [131], where they show that a RNN with transition matrix initialised to the identity, with biases initialized to zero, can 'remember' information over a long period using ReLU non-linearities. Saxe et al. [190] study exact solutions to learning dynamics in deep networks and find that orthogonal weight initializations at each layer lead to depth-independent learning (thus escaping the vanishing/exploding gradient problem). Interestingly, they attribute this to the eigenvalue spectrum of orthogonal matrices lying on the unit circle. They compare with weights initialized to random, scaled Gaussian values, which preserve norms in expectation (over values of the random matrix) and find orthogonal matrices superior. It therefore appears that preserving norms is *not* sufficient to stabilize gradients over network depth, but that the eigenvalue spectrum must also be strictly controlled. Henaff et al. [90] also study analytic solutions to the long-term memory task, with results supporting observations and intuitions that orthogonal (or unitary) matrices would be appropriate as transition matrices for this task. They also study initializations to orthogonal and identity matrices, and consider experiments where an additional term in the loss function encourages an orthogonal solution to the transition matrix without using an explicit parametrization.

This approach of using a soft constraint is also used in Mikolov et al. [154], where part of the transition matrix is constrained to be close to the identity.

In a related but separate vein, Krueger and Memisevic [128] penalize the difference of difference of norms between subsequent hidden states in the network. It is however not equivalent to imposing orthogonality of the *transition* matrix, as the norm of the hidden state may be influenced by the inputs and non-linearities.

Hard constraint through parametrisation

Rather than introducing penalties in the loss function for transition matrices deviation from unitarity, we can rather constrain the parameter space to enforce it. In this case, we can perform gradient updates in the *parameter space*, and so long as it is closed under addition, we always retain a unitary transition matrix. The contribution in this chapter lies in this realm.

We draw most inspiration from the work of Arjovsky et al. [6]. Citing the difficulty of obtaining a general and efficient parametrization of unitary matrices, they use the fact that the unitary group is closed under matrix multiplication to form a composite operator:

$$U = \mathbf{D}_3\mathbf{R}_2\mathcal{F}^{-1}\mathbf{D}_2\Pi\mathbf{R}_1\mathcal{F}\mathbf{D}_1 \tag{3.7}$$

where each component is unitary and parametrized as follows:

- $\mathbf{D}$ is a diagonal matrix with entries of the form $e^{i\alpha}$, $\alpha \in \mathbb{R}$
- $\mathbf{R}$ is a complex reflection operator; $\mathbf{R} = \mathbb{I} - 2\frac{\mathbf{v}\mathbf{v}^\dagger}{\|\mathbf{v}\|^2}$ († denotes Hermitian conjugate)

- $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and inverse Fourier transforms (or, in practice, their discrete matrix representations)
- Π is a fixed permutation matrix

In total, this parametrization has $7n$ real learnable parameters ($2n$ for each reflection and n for each diagonal operator), and so must describe a subspace of unitary matrices (which have n^2 real parameters) for $n > 7$. Nonetheless, they find that an RNN using this operator as its transition matrix outperforms LSTMs on the adding and memory tasks described first in Hochreiter and Schmidhuber [95], which are described in detail in Sections 3.4.1 and 3.4.2.

This prompted us to consider other parametrizations of unitary matrices which might be more expressive or interpretable. In particular, we exploit the fact that the unitary group is a Lie group to devise a fully general parametrisation in terms of the Lie algebra (see Section 3.2.2).

Since the publication of the work described in this chapter, other approaches to parametrise orthogonal or unitary RNNs have been developed. For example, Mhammedi et al. [150] represent orthogonal matrices as products of Householder reflections, noting that it is sufficient to use higher-dimensional orthogonal matrices rather than complex-valued unitary matrices. More recently, Helfrich et al. [89] uses what they call a scaled Cayley transformation to represent orthogonal matrices, and describe an update scheme based on the gradients of this transformation.

3.2 Parametrising a unitary RNN

Since the proposed parametrisation relies on the fact that unitary matrices are elements of a Lie group, I first give some background on unitary matrices, their group structure, and Lie groups and algebras.

3.2.1 Lie groups and lie algebras

Background: The unitary group

Unitary matrices are (potentially) complex, square matrices U such that

$$U^\dagger U = UU^\dagger = \mathbb{I} \tag{3.8}$$

where $U^\dagger = (U^*)^T$ is the Hermitian conjugate of U.

As such, unitary matrices generalise orthogonal matrices (with $O^T O = OO^T = \mathbb{I}$) to the complex field.

The unitary property (Equation 3.8) implies the norm-preserving property of unitary operators. If $\mathbf{x} \in \mathbb{C}^d$ is a d-dimensional (potentially) complex vector[4], then

$$\|U\mathbf{x}\| = (U\mathbf{x})^\dagger U\mathbf{x} = \mathbf{x}^\dagger U^\dagger U\mathbf{x} = \mathbf{x}^\dagger\mathbf{x} = \|\mathbf{x}\| \tag{3.9}$$

We additionally have that all eigenvalues of U lie on the unit circle in $\mathbb{C}$.

This is easy to see: suppose $\mathbf{v}$ is an eigenvector of U with eigenvalue λ, so

$$U\mathbf{v} = \lambda\mathbf{v}$$

[4]If $\mathbf{x}$ is purely real, than all $\dagger$ operations are equivalent to transposition.

Taking the norm on both sides:

$$|U\mathbf{v}| = |\lambda\mathbf{v}|$$

However, since U is unitary

$$|U\mathbf{v}| = |\mathbf{v}|$$
$$\Rightarrow |\lambda| = 1$$

This means that, if $\lambda \in C$, it must be of the form $\lambda = e^{i\theta}$ for some $\theta \in \mathbb{R}$, thus it lies on the unit circle. This also implies that the determinant of a unitary matrix lies on the unit circle.

Next, we can show that unitary matrices form a group, with group operation matrix multiplication.

- The identity element is $\mathbb{I}$, which is trivially unitary.
- All unitary matrices U have an inverse, which is $U^\dagger$.
- The group is closed under multiplication: if U and V are unitary, then UV is also unitary:
$$(UV)^\dagger UV = V^\dagger U^\dagger UV = V^\dagger \mathbb{I} V = \mathbb{I}$$

We refer to the group of $n \times n$ unitary matrices as as $U(n)$. Orthogonal $n \times n$ matrices form a subgroup $O(n)$ of $U(n)$.

Lie groups and Lie algebras

An n-dimensional manifold is a metric space which 'looks' locally similar to $\mathbb{R}^n$. That is, at any point in the manifold, the neighbourhood of that point is

homeomorphic to $\mathbb{R}^n$. A *differentiable* manifold is a manifold with additional structure which allows calculus on the manifold. Further details are beyond the current scope, see Spivak [199] for an exhaustively comprehensive review of differentiable manifolds and other structures.

A *Lie group* then is a group which is also a differentiable manifold, with elements of the group corresponding to points on the manifold. The group operations (multiplication and inversion):

$$\begin{aligned} (x, y) &\mapsto xy \\ x &\mapsto x^{-1} \end{aligned} \tag{3.10}$$

must be infinitely differentiable. The unitary group $U(n)$ is a Lie group, where its group operation is matrix multiplication.

The differentiable manifold property of Lie groups opens the door for the study of the Lie *algebra*. This object is the *tangent space* to the Lie group at the identity (the group must have an identity element).

Consider a curve through the unitary group $U(n)$ - a one-dimensional subspace parametrized by a variable t, where $U(t = 0) = \mathbb{I}$ (this is a matrix $U(t)$ in $U(n)$ parametrised by t, not a group).

Consider the defining property of unitary matrices (Equation 3.8), and take the derivative along this curve:

$$U(t)^\dagger U(t) = \mathbb{I} \rightarrow \dot{U}(t)^\dagger U(t) + U^\dagger(t)\dot{U}(t) = 0 \tag{3.11}$$

Taking $t \rightarrow 0$, $U(t) \rightarrow \mathbb{I}$, we have

$$\dot{U}(0)^\dagger \mathbb{I} + \mathbb{I}^\dagger \dot{U}(0) = 0 \Rightarrow \dot{U}(0)^\dagger = -\dot{U}(0) \tag{3.12}$$

The elements $\dot{U}(0)$ belong to the Lie algebra. We refer to this Lie algebra as $\mathfrak{u}(n)$, and an arbitrary element as L. Then Equation 3.12 defines the properties of these Lie algebra elements; they are $n \times n$ skew-Hermitian matrices: $L^\dagger = -L$.

As vector spaces, Lie algebras are closed under addition. In particular $\mathfrak{u}(n)$ is a vector space over $\mathbb{R}$, so a *real* linear combination of its elements is once again in $\mathfrak{u}(n)$ (this is also clear from the definition of skew-Hermitian). We exploit this fact to build the parametrisation closed under (additive) gradient updates.

Lie algebras are interesting algebraic objects and have been studied deeply (see e.g. Spivak [199] for a review), but in this work we use $\mathfrak{u}(n)$ because of the *exponential map*.

Above, it was shown that elements of the algebra can be derived from the group, considering infinitesimal steps away from the identity. There is a reverse operation, allowing elements of the group to be recovered from the algebra: this is the *exponential map*. In the case of matrix groups, the exponential map is simply the *matrix exponential*:

$$\exp(L) = \sum_{j=0}^{\infty} \frac{L^j}{j!} \tag{3.13}$$

Very simply, if $L \in \mathfrak{u}(n)$, then $\exp(L) \in U(n)$. While this map is not in *general* surjective, it so happens that $U(n)$ is a compact, connected group and so $\exp$ *is* indeed surjective [210]. That is, for any $U \in U(n)$, there exists *some* $L \in \mathfrak{u}(n)$ such that $\exp(L) = U$.

While orthogonal matrices also form a Lie group $O(n)$, with associated Lie algebra $\mathfrak{o}(n)$ consisting of skew-symmetric matrices, $O(n)$ is *not* connected. $O(n)$ consists of two connected components corresponding to orthogonal matrices

with determinant -1 and 1. The exponential map can only produce *special* orthogonal matrices - those with determinant one, since this component of $O(n)$ contains its identity element. This division does not exist in the unitary group: all determinants lying on the unit circle are allowed (there is a continuous route from 1 to -1), so it is connected.

Lie groups in machine learning

The theory of Lie groups and Lie algebras has seen most application in machine learning for its use in capturing notions of *invariance*. For example, Miao and Rao [151], learn infinitesimal Lie group generators (elements of the Lie algebra) associated with affine transformations of images, corresponding to visual perceptual invariances. This is different to our setting as our generators are already known (we assume the Lie group $U(n)$) and wish to learn the coefficients of a given transformation relative to that basis set of generators. However, our approach could be extended to the case where the basis of $\mathfrak{u}(n)$ is *unknown*, and must be learned. As we find later (Section 3.3.5), the choice of basis can impact performance, and so may be an important consideration. Cohen and Welling [42] learn commutative subgroups of $SO(n)$ (known as toroidal subgroups), motivated by learning the irreducible representations of the symmetry group corresponding to invariant properties of images. Their choice of group parametrization is equivalent to selecting a particular basis of the corresponding Lie algebra, as they describe, but primarily exploit the algebra to understand properties of toroidal subgroups.

Tuzel et al. [216] perform motion estimation by defining a regression function in terms of a function on the Lie algebra of affine transformations, and then

learning this. This is similar to our approach in the sense that they do optimization in the Lie algebra, although as they consider two-dimensional affine transformations only, their parametrization of the Lie algebra is straight forward.

3.2.2 Parametrization of $U(n)$ in terms of $\mathfrak{u}(n)$

We now have the mathematical machinery to define the parametrisation. It should be noted that this relationship between the Lie group and the Lie algebra has been known in mathematics for a long time. The contribution in this work is to exploit it for use in machine learning, specifically for unitary RNNs.

The idea is simple: given a basis of $\mathfrak{u}(n)$, we can define any element L of $\mathfrak{u}(n)$ by its coordinates $\{\lambda_j\}$ under this basis. Since the exponential map between $\mathfrak{u}(n)$ is surjective, given any U in $U(n)$ there exists an L, and thus coordinates $\{\lambda_j\}$ under the basis, which define U. Note that since $U(n)$ is compact and connected, the exponential map is surjective (but not injective), so we are guaranteed to find an L which maps to U. However, this L is not guaranteed to be unique.

In detail: The dimension of $\mathfrak{u}(n)$ as a real vector space is n^2. This is readily derived from noting that an arbitrary $n \times n$ complex matrix has $2n^2$ free real parameters, and the requirement of $L^\dagger = -L$ imposes n^2 constraints. So, a set of n^2 linearly-independent skew-Hermitian matrices defines a basis for the space; $\{T_j\}_{j=\{1,\dots,n^2\}}$. Then any element L can be written as

$$L = \sum_{j=1}^{n^2} \lambda_j T_j \tag{3.14}$$

where $\{\lambda_j\}_{j=1,\dots,n^2}$ are n^2 real numbers; the coefficients of L with respect to the basis.

Using the exponential map,

$$U = \exp(L) = \exp\left(\sum_{j=1}^{n^2} \lambda_j T_j\right) \tag{3.15}$$

we see that these $\{\lambda_j\}_{j=1,\dots,n^2}$ suffice as *parameters* of U (given the basis T_j).

This is the parametrization we propose.

It has two attractive properties:

1. It is a fully 'general' parametrization, as the exponential map is surjective
2. Gradient updates on $\{\lambda_j\}_{j=1,\dots,n^2}$ preserve unitarity automatically, as the algebra is closed under addition

This parametrization means gradient steps are taken in the vector space of $\mathfrak{u}(n)$, rather than the manifold of $U(n)$.

There are many possible choices of basis for $\mathfrak{u}(n)$.

We choose the following set of n^2 sparse matrices:

1. n diagonal, imaginary matrices: T_a is i on the a-th diagonal, else zero.
2. $\frac{n(n-1)}{2}$ symmetric, imaginary matrices with two non-zero elements
3. $\frac{n(n-1)}{2}$ anti-symmetric, real matrices with two non-zero elements

The effects of this choice are exploded in Section 3.3.5.

For clarity, we look at a concrete example where $n = 2$. The basis has the following elements:

$$T_1 = \begin{pmatrix} i & 0 \\ 0 & 0 \end{pmatrix} \qquad T_2 = \begin{pmatrix} 0 & 0 \\ 0 & i \end{pmatrix} \qquad T_3 = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \qquad T_4 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{3.16}$$

Then by our parametrisation (Equation 3.15), the element $U_a \in U(n)$ with parameters $(a, 0, 0, 0)$ ($a \in \mathbb{R}$) is:

$$U_a = \exp(aT_1) = \exp(\text{diag}(ai, 0, 0, 0)) = \begin{pmatrix} e^{ia} & 0 \\ 0 & 1 \end{pmatrix} \tag{3.17}$$

Which is easily seen to be unitary.

More generally, we have $U_u \in U(n)$ with parameters (a, b, c, d) (all in $\mathbb{R}$):

$$U_u = \exp(aT_1 + bT_2 + cT_3 + dT_4) = \exp \begin{pmatrix} ai & ci + d \\ ci - d & bi \end{pmatrix} \tag{3.18}$$

The determinant of U_u is

$$\det(U_u) = \exp \text{Tr} \begin{pmatrix} ai & ci + d \\ ci - d & bi \end{pmatrix} = \exp((a + b)i) \tag{3.19}$$

which has modulus 1 since $a + b \in \mathbb{R}$, as expected.

We can see that U_u is unitary by noting that any exponentiated skew-Hermitian matrix is unitary. Suppose the skew-Hermitian matrix is L, with $U = \exp(L)$. Then

$$U^\dagger U = \exp(L)^\dagger \exp(L) = \exp(L^\dagger) \exp(L) = \exp(L^\dagger + L) = \exp(-L + L) = \exp(0) = \mathbb{I}$$

This follows because $L^\dagger$ commutes with L, and we have $\exp(A)\exp(B) = \exp(A + B)$ if A and B commute. Then, we use that L is skew-Hermitian.

3.2.3 Derivatives of the matrix exponential

The matrix exponential appearing in Equation 3.15 poses an issue for gradient calculations. In general, the derivative of the matrix exponential does not have

a closed-form expression, so computing gradients is intractable. In this section, we describe a mathematical trick to address the computational complexity issue. This is based on work from Kalbfleisch and Lawless [113] and Jennrich and Bright [106], and exploits our choice of basis.

Exploiting the fact that L is skew-Hermitian, we can derive an analytical expression for the derivative of U with respect to each of its parameters.

This expression takes the form:

$$\frac{\partial U}{\partial \lambda_a} = W V_a W^\dagger \tag{3.20}$$

where W is a unitary matrix of eigenvectors obtained in the eigenvalue decomposition of U; $U = WDW^\dagger$, (D = diag(d_1, ..., d_{n^2}); d_i are the eigenvalues of U).

Each V_a is a matrix defined component-wise

$$i = j : V_{ii} = (W^\dagger T_a W)_{ii} e^{d_i} \tag{3.21}$$

$$i \neq j : V_{ij} = (W^\dagger T_a W)_{ij} \left(\frac{e^{d_i} - e^{d_j}}{d_i - d_j} \right) \tag{3.22}$$

Where T_a is the basis matrix of the Lie algebra in the a-th direction.

We can simplify the expression $W^\dagger T_a W$ for each T_a, depending on the type of basis element. In these expressions, $\mathbf{w}_a$ refers to the a-th *row* of W.

1. T_a purely imaginary; $W^\dagger T_a W = i \cdot \text{outer}(\mathbf{w}_a^*, \mathbf{w}_a)$
2. T_a symmetric imaginary, non-zero in positions (r, s) and (s, r): $W^\dagger T_{rs} W = i \cdot (\text{outer}(\mathbf{w}_s^*, \mathbf{w}_r) + \text{outer}(\mathbf{w}_r^*, \mathbf{w}_s))$
3. T_a antisymmetric real, non-zero in positions (r, s) and (s, r): $W^\dagger T_{rs} W = \text{outer}(\mathbf{w}_r^*, \mathbf{w}_s) - \text{outer}(\mathbf{w}_s^*, \mathbf{w}_r)$

These expressions follow from the sparsity of the basis and are derived in section 3.2.4. Thus, we reduce the calculation of $W^\dagger T_a W$ from two matrix multiplications to at most two vector outer products. Overall, this reduces the cost of calculating gradients to a single eigenvalue decomposition, and for each parameter two matrix multiplications (equation 3.20), one or two vector outer products, and element-wise multiplication of two matrices (equations 3.21, 3.22). This makes computing gradients of the exponential map (for this choice of basis) computationally practical.

3.2.4 Derivative of the matrix exponential: details

In this section, we derive the expressions above.

We have $U = \exp(L)$, and seek dU. For what follows, we simply require that L be normal ($L^\dagger L = LL^\dagger$), so the results are more general than the unitary case.

In this case, L is skew-Hermitian, which is normal, since $L^\dagger L = \mathbb{I} = LL^\dagger$. Normal matrices are diagonalisable by unitary matrices. Thus, there exist $W \in U(n)$ and $D = \mathrm{diag}(d_1, \ldots, d_n)$ such that $L = WDW^\dagger$, and therefore

$$U = W\tilde{D}W^\dagger \tag{3.23}$$

where $\tilde{D} = \mathrm{diag}(e^{d_1}, \ldots, e^{d_n})$.

We assume we can calculate: dL, W, and D and seek an expression for dU.

Then using 3.23:

$$\begin{aligned} dU &= d(W\tilde{D}W^\dagger) \\ &= dW\tilde{D}W^\dagger + Wd\tilde{D}W^\dagger + W\tilde{D}dW^\dagger \end{aligned} \tag{3.24}$$

Pre-multiplying with $W^\dagger$ and post-multiplying with W:

$$W^\dagger dUW = W^\dagger dW\tilde{D} + d\tilde{D} + \tilde{D}dW^\dagger W \tag{3.25}$$

The last term can be simplified by differentiating both sides of $W^\dagger W = \mathbb{I}$ (this follows from unitarity of W);

$$W^\dagger W + W^\dagger dW = 0 \Rightarrow dW^\dagger W = -W^\dagger dW \tag{3.26}$$

and substituting back into 3.25 to get:

$$W^\dagger dUW = W^\dagger dW\tilde{D} - \tilde{D}W^\dagger dW + d\tilde{D} \tag{3.27}$$

We can then say that $dU = WVW^\dagger$ where

$$V = W^\dagger dW\tilde{D} - \tilde{D}W^\dagger dW + d\tilde{D} \tag{3.28}$$

Similarly, $dL = WAW^\dagger$ where (replacing $\tilde{D}$ with D)

$$A = W^\dagger dWD - DW^\dagger dW + dD \tag{3.29}$$

and also $A = W^\dagger dLW$.

Calculating V

We use the convention that repeated indices denote summation over that index, unless otherwise stated.

Looking at the components of V;

$$V_{ij} = (W^\dagger dW\tilde{D})_{ij} - (\tilde{D}W^\dagger dW)_{ij} + d\tilde{D}_{ij} \tag{3.30}$$

Diagonal case ($i = j$): (no summation over i)

$$V_{ii} = W^{\dagger}_{ia} dW_{ab} \tilde{D}_{bi} - \tilde{D}_{ia} W^{\dagger}_{ab} dW_{bi} + d\tilde{D}_{ii} \tag{3.31}$$

Since $\tilde{D}_{bi} = \delta_{bi}\tilde{d}_i$, the first two terms cancel:

$$\begin{aligned} V_{ii} =& W^{\dagger}_{ia} dW_{ab} \delta_{bi} \tilde{d}_i - \delta_{ai} \tilde{d}_i W^{\dagger}_{ab} dW_{bi} + d\tilde{D}_{ii} \\ =& W^{\dagger}_{ia} dW_{ai} \tilde{d}_i - \tilde{d}_i W^{\dagger}_{ib} dW_{bi} + d\tilde{D}_{ii} \\ =& d\tilde{D}_{ii} \end{aligned} \tag{3.32}$$

Using 3.29 we get $A_{ii} = dD_{ii} = (W^{\dagger} dL W)_{ii}$

Recall that the diagonal elements of $\tilde{D}$ are the exponentiated versions of the diagonal elements of D, so $\tilde{D}_{ii} = e^{d_i}$.

Then

$$d\tilde{D}_{ii} = d(d_i) e^{d_i} = dD_{ii} \tilde{D}_{ii} \tag{3.33}$$

Inserting that into Equation 3.32:

$$V_{ii} = dD_{ii} \tilde{D}_{ii} = (W^{\dagger} dL W)_{ii} \tilde{D}_{ii} = (W^{\dagger} dL W)_{ii} e^{d_i} \tag{3.34}$$

This produces Equation 3.21.

Off-diagonal case ($i \neq j$): (no summation over i, j) In this case, the purely diagonal part vanishes. We get:

$$\begin{aligned} V_{ij} =& W^{\dagger}_{ia} dW_{ab} \delta_{bj} \tilde{d}_j - \delta_{ai} \tilde{d}_i W^{\dagger}_{ab} dW_{bj} \\ =& W^{\dagger}_{ia} dW_{aj} \tilde{d}_j - W^{\dagger}_{ib} dW_{bj} \tilde{d}_i \\ =& (W^{\dagger} dW)_{ij} (\tilde{d}_j - \tilde{d}_i) \end{aligned} \tag{3.35}$$

Similarly,

$$A_{ij} = (W^{\dagger} dW)_{ij} (d_j - d_i) \tag{3.36}$$

Remembering that this is all component-wise multiplication (no summation over i and j), we can rearrange expressions to get:

$$(W^\dagger dW)_{ij} = \frac{A_{ij}}{d_j - d_i} = \frac{(W^\dagger dLW)_{ij}}{d_j - d_i} \tag{3.37}$$

Combining this with 3.35 and remembering $\tilde{d}_a = e^{d_a}$, we have, for $i \neq j$:

$$V_{ij} = (W^\dagger dLW)_{ij} \left(\frac{e^{d_i} - e^{d_j}}{d_i - d_j} \right) \tag{3.38}$$

This is Equation 3.22.

Efficiently calculating $W^\dagger dLW$

This section is specific to our work, as it relies on the choice of basis for $\mathfrak{u}(n)$.

In our case, dL is simple. L is a linear combination of the parameters λ_i;

$$L = \sum_i^{n^2} \lambda_i T_i \tag{3.39}$$

Where T_i are the basis matrices of $\mathfrak{u}(n)$.

Then

$$dL_a = \frac{\partial L}{\partial \lambda_a} = T_a \tag{3.40}$$

We need $W^\dagger T_a W$ for all a. Since the T_as are sparse, this is cheaper than performing n^2 full matrix multiplications, as we demonstrate now. Cases:

T_a diagonal, purely imaginary

T_a is zero except for a i in the a-th position on the diagonal.

$$\begin{aligned} (W^\dagger T_a W)_{ij} &= i W^\dagger_{ia} W_{aj} = i W^*_{ai} W_{aj} \\ \Rightarrow W^\dagger T_a W &= i \cdot \mathrm{outer}(\mathbf{w}^*_a, \mathbf{w}_a) \end{aligned} \tag{3.41}$$

where $\mathbf{w}_a$ is the a-th *row* of W.

T_a symmetric, purely imaginary

T_{rs} is zero except for i in position (r, s) and (s, r).

$$\begin{aligned}(W^\dagger T_{rs} W)_{ij} &= i W^\dagger_{ik}(\delta_{ks,lr} + \delta_{kr,ls}) W_{lj} \\ &= i(W^\dagger_{is} W_{rj} + W^\dagger_{ir} W_{sj}) = i(W^*_{si} W_{rj} + W^*_{ri} W_{sj}) \\ \Rightarrow W^\dagger T_{rs} W &= i \cdot (\text{outer}(\mathbf{w}^*_s, \mathbf{w}_r) + \text{outer}(\mathbf{w}^*_r, \mathbf{w}_s))\end{aligned} \tag{3.42}$$

T_a antisymmetric, purely real

T_{rs} is zero except for 1 in position (r, s) and -1 in position (s, r).

$$\begin{aligned}(W^\dagger T_{rs} W)_{ij} &= W^\dagger_{ik}(\delta_{kr,sl} - \delta_{ks,rl}) W_{lj} \\ &= W^\dagger_{ir} W_{sj} - W^\dagger_{is} W_{rj} = W^*_{ri} W_{sj} - W^*_{si} W_{rj} \\ \Rightarrow W^\dagger T_{rs} W &= \text{outer}(\mathbf{w}^*_r, \mathbf{w}_s) - \text{outer}(\mathbf{w}^*_s, \mathbf{w}_r)\end{aligned} \tag{3.43}$$

These reproduce the expressions at the end of section 3.2.3. The outer product of two n-dimensional vectors is an $O(n^2)$ operation, and so this provides a (up to) factor n speed-up on matrix multiplication.

3.3 Supervised learning of unitary matrices

Before attempting to use the parametrisation to learn unitary RNNs, we first demonstrate that the parametrisation can be used to learn unitary matrices in general.

We consider the supervised learning problem of learning the unitary matrix U that generated a $\mathbf{y}$ from $\mathbf{x}$; $\mathbf{y} = U\mathbf{x}$, given examples of such $\mathbf{x}$s and $\mathbf{y}$s. This is the core learning problem that needs to be solved for the state-transformation matrix in RNNs. It is similar to the setting considered in Hazan et al. [87] (they consider an online learning problem).

We compare a number of methods for learning U at different values of n. We further consider the case where we have artificially restricted the number of learnable variables in our parametrization (for the sake of comparison; section 3.3.3, and generate a pathological change of basis to demonstrate the relevance of selecting a good basis (section 3.3.5).

3.3.1 Task

The experimental setup is as follows: we create a $n \times n$ unitary matrix U (the next section describes how this is done), then sample vectors $\mathbf{x} \in \mathbb{C}^n$ with normally-distributed coefficients. We create $\mathbf{y}_j = U\mathbf{x}_j + \epsilon_j$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

The objective is to recover U from the $\{\mathbf{x}_j, \mathbf{y}_j\}$ pairs by minimizing the squared Euclidean distance between predicted and true $\mathbf{y}$ values;

$$U = \underset{U}{\operatorname{argmin}} \frac{1}{N} \sum_{j}^{N} \|\hat{\mathbf{y}}_\mathbf{j} - \mathbf{y}_\mathbf{j}\|^2 = \underset{U}{\operatorname{argmin}} \frac{1}{N} \sum_{j}^{N} \|U\mathbf{x}_j - \mathbf{y}_\mathbf{j}\|^2 \tag{3.44}$$

While this problem is easily solved in the batch setting using least-squares, we wish to learn U through mini-batch stochastic gradient descent, to emulate a deep learning scenario.

For each experimental run (a single U), we generate one million training $\{\mathbf{x}_j, \mathbf{y}_j\}$ pairs, divided into batches of size 20. The test and validation sets both

contain $100,000$ examples. In practice we set $\sigma^2 = 0.01$ and use a fixed learning rate of 0.001. For larger dimensions, we run the model through the data for multiple epochs, shuffling and re-batching each time.

All experiments were implemented in Python. For the matrix exponential, we use the scipy built-in `expm`, which uses Pade approximation [3]. We make use of the fact that iL is Hermitian to use `eigh` (also in scipy) to perform eigenvalue decompositions.

Generating the ground-truth unitary matrix The U we wish to recover is generated by one of three methods:

1. QR decomposition: we create a $n \times n$ complex matrix with normally-distributed entries and then perform a QR decomposition, producing a unitary matrix U and an upper triangular matrix (which is discarded). This approach is also used to sample orthogonal matrices in Hazan et al. [87], noting a result from Stewart [204] demonstrating that this is equivalent to sampling from the appropriate Haar measure.
2. Lie algebra: given the standard basis of $\mathfrak{u}(n)$, we sample n^2 normally-distributed real λ_j to produce $U = \exp\left(\sum_j \lambda_j T_j\right)$
3. Unitary composition: we compose parametrized unitary operators as in Arjovsky et al. [6] (Equation 3.7). The parameters are sampled as follows: angles in D come from $\mathcal{U}(-\pi, \pi)$. The complex reflection vectors in $\mathbf{R}$ come from $\mathcal{U}(-s, s)$ where $s = \sqrt{\frac{6}{2n}}$.

We study the effects of this generation method on test-set loss in section 3.3.4. While we find no significant association between generation method and learn-

ing approach, in the experiments we nonetheless average over an equal number of experiments using each method, to compensate for possible unseen bias.

Learning approaches We compare the following approaches for learning U:

1. `projection`: U is represented as an unconstrained $n \times n$ complex matrix, but after each gradient update we *project* it to the closest unitary matrix, using polar decomposition [115]. This amounts to $2n^2$ real parameters.
2. `arjovsky`: U is parametrized as in Equation 3.7, which comes to $7n$ real parameters.
3. `lie_algebra`: (we refer to this as $\mathfrak{u}(n)$) U is parametrized by its n^2 real coefficients $\{\lambda_j\}$ in the Lie algebra, as in Equation 3.15.

As baselines we use the `true` matrix U, and a `random` unitary matrix U_R generated by the same method as U (in that experimental run).

We also implemented the algorithm described in Hazan et al. [87] and considered both unitary and orthogonal learning tasks (our parametrization contains orthogonal matrices as a special case) but found it too numerically unstable and therefore excluded it from our analyses.

3.3.2 Comparison of approaches

Table 3.1 shows the test-set loss for different values of n and different approaches for learning U. We performed between 6 and 18 replicates of each experiment, and show bootstrap estimates of means and standard errors over

Table 3.1: **Performance of different parametrisations in a supervised unitary matrix learning task**. The table shows the mean l_2-norm between $\hat{y}_i$ and y_i on the test set for the different approaches as the dimension of the unitary matrix changes. `true` refers to the matrix used to generate the data, `projection` is the approach of 're-unitarising' using a polar decomposition after gradient updates, `arjovsky` is the composition approach defined in Equation 3.7, $\mathfrak{u}(n)$ is our parametrization (Equation 3.15) and `rand` is a random unitary matrix generated in the same manner as `true`. Values in bold are the best for that n (excluding `true`). The error for `true` is typically very small, so we omit it.

n	`true`	`projection`	`arjovsky`	`lie algebra`	`rand`
3	$6.004 \pm 0.005 \times 10^{-4}$	8 ± 1	$\mathbf{6.005 \pm 0.003 \times 10^{-4}}$	$\mathbf{6.003 \pm 0.003 \times 10^{-4}}$	12.5 ± 0.4
6	~ 0.001	15 ± 1	0.09 ± 0.01	$\mathbf{0.03 \pm 0.01}$	24 ± 1
8	~ 0.002	14 ± 1	1.17 ± 0.06	$\mathbf{0.014 \pm 0.006}$	31.6 ± 0.6
14	~ 0.003	24 ± 4	10.8 ± 0.3	$\mathbf{0.07 \pm 0.02}$	52 ± 1
20	~ 0.004	38 ± 3	29.0 ± 0.5	$\mathbf{0.47 \pm 0.03}$	81 ± 2

these replicates. As we can see, the learning task becomes more challenging as n increases, but the parametrization using $\mathfrak{u}(n)$ consistently achieves low error and outperforms the other approaches. In the next sections, we explore some properties and choices of the parametrisation.

3.3.3 Restricting to $7n$ parameters

As mentioned, the method `arjovsky` uses only $7n$ parameters. To check if this difference accounts for the differences in loss observed in Table 3.1, we ran experiments where we fixed all but $7n$ (selected randomly) of the $\{\lambda_j\}$ in the `lie_algebra` parametrization. The fixed parameters retained their initial values throughout the experiment. We observe that, as suspected, restricting to $7n$ parameters results in a performance degradation equivalent to that of `arjovsky`.

Table 3.2 shows the results for $n = 8, 14, 20$. The fact that the restricted case

Table 3.2: **Impact of restricting the parametrisation.** We observe that restricting our approach to the same number of learnable parameters as that of Arjovsky and Shah (2015) causes a similar degradation in performance on the task. This indicates that the relatively superior performance of our model is explained by its generality in capturing arbitrary unitary matrices.

n	`arjovsky`	`lie_restricted`	`lie_unrestricted`
8	1.2 ± 0.1	1.0 ± 0.2	0.04 ± 0.01
14	11.6 ± 0.3	12.6 ± 0.4	0.25 ± 0.03
20	27.8 ± 0.7	28.0 ± 0.6	0.19 ± 0.03

is consistently within error of the `arjovsky` model supports our hypothesis that the difference in learnable parameters accounts for the difference in performance. This suggests that generalising the model of Arjovsky et al. [6] to allow for n^2 parameters may result in performance similar to our approach. However, how to go about such a generalisation is unclear, as a naive approach would simply use a composition of n^2 operators, and this would likely become computationally intractable.

3.3.4 Method of generating U

As described, we used three methods to generate the true U. One of these produces U in the subspace available to the composition parametrization (Equation 3.7), so we were curious to see if this parametrization performed better on experiments using that method. We were also concerned that generating U using the Lie algebra parametrization might make the task too 'easy' for our approach, as its random initialization could lie close to the true solution.

Figure 3.2 shows box-plots of the distribution of test losses from these approaches for the three methods, comparing our approach ($\mathfrak{u}(n)$) with that of

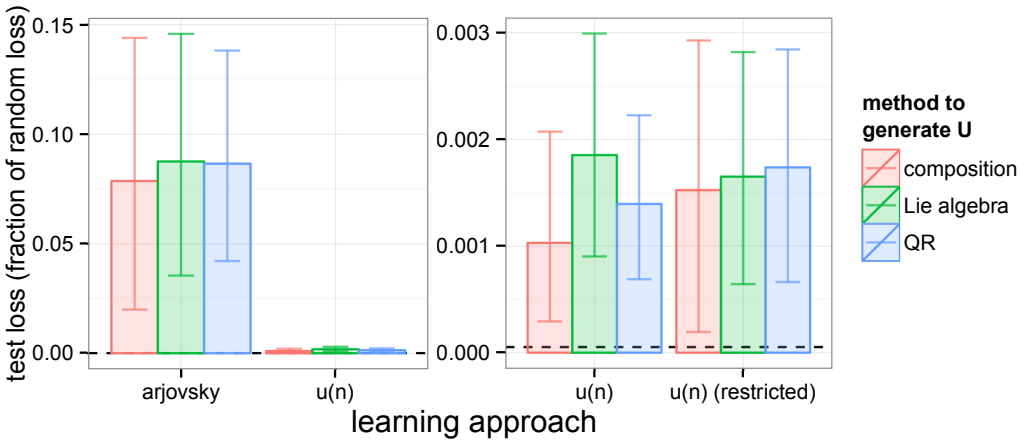


Figure 3.2: **Impact of method of generating the unitary matrix.** We ask whether the method used to generate U influences performance for different approaches to *learning* U. Error bars are bootstrap estimates of 95% confidence intervals. To compare across different n's, we normalise each loss by the loss of `rand` for that n, and report fractions. The dotted line is the `true` loss, similarly normalised. the choice of method to generate U does not appear to affect test-set loss for the different approaches. *Right*: Finer resolution on the $\mathfrak{u}(n)$ result in left panel. We also include the case where we restrict to $7n$ learnable parameters.

Arjovsky et al. [6], denoted `arjovsky`. To combine results from experiments using different values of n, we first scaled test-set losses by the performance of `rand` (the random unitary matrix), so the y-axis ranges from 0 (perfect) to 1 (random performance). The dotted line denotes the average (over methods) of the test-set loss for `true`, similarly scaled. The right panel in Figure 3.2 shows a zoomed-in version of the $\mathfrak{u}(n)$ result where the comparison with `true` is more meaningful, and a comparison with the case where we have restricted to $7n$ learnable parameters (see section 3.3.3).

We do not observe a difference (within error) between the methods, which is consistent between $\mathfrak{u}(n)$ and `arjovsky`. Our concern that using the Lie algebra to generate U would make the task 'too easy' for $\mathfrak{u}(n)$ was seemingly unfounded.

3.3.5 Changing the basis of $\mathfrak{u}(n)$

The Lie group parametrization assumes a fixed basis of $\mathfrak{u}(n)$. Our intuition is that this makes some regions of $U(n)$ more 'accessible' to the optimization procedure, elements whose coefficients are small given this basis. Learning a matrix U which came from elsewhere in $U(n)$ may therefore be more challenging. We emulated this 'change of basis' without needing to explicitly construct a new basis by generating a change of basis matrix, M. That is, if V_j is the j-th element of the new basis, it is given by

$$V_j = \sum_k M_{jk} T_k \tag{3.45}$$

If $\{\tilde{\lambda}\}_a$ are the coefficients of L relative to the basis V, the coefficients relative to the old basis T are given by:

$$\lambda_b = \sum_k \tilde{\lambda}_k M_{kb} = \tilde{\lambda}^T \cdot M \tag{3.46}$$

A change of basis matrix must be full-rank. We generate one by sampling a square, $n^2 \times n^2$ matrix from a continuous uniform distribution $\mathcal{U}(-c, c)$ (c is a constant we vary in experiments, see Figure 3.3). This is very unlikely to be singular. We choose the c range of the distribution such that M will have 'large' values relative to the true matrix U, whose parameters λ (relative to T) are drawn from $\mathcal{N}(0, 0.01)$.

Preliminary experiments suggested that the learning rate must be adjusted to compensate for the change of scale - evidence for this is visible in the first column of Figure 3.3, where changing the basis without changing the learning rate results in an unstable validation set trace. Poor performance resulting from an inappropriate learning rate is not our focus here, so we performed experiments

for different values of the learning rate. Figure 3.3 shows a grid of validation set losses as we vary the learning rate (columns) and the value of c (rows).

Our intuition is that if the performance under the change of basis is *purely* driven by the difference in scale, using an appropriately-scaled learning rate should negate its affect. Each parameter λ_j is scaled by a variable uniformly distributed between $(-c, c)$. The expectation value of the absolute value of this quantity is $c^2/2$, so we consider learning rates normalised by this factor.

As seen in Figure 3.3, the graphs on the diagonal are *not* identical, suggesting that merely scaling the learning rate does not account for the change of learning behaviour given a new basis - at least in expectation. Nonetheless, it is reassuring to observe that for all choices of c explored, there exists a learning rate which facilitates learning, even if it markedly slower than the 'ideal' case. While having a 'misspecified' basis does appear to negatively impact learning, it can be largely overcome with choice of learning rate.

3.4 Unitary RNN for long memory tasks

The results from the previous section indicate that the parametrisation using the Lie algebra allows us to learn unitary matrices. Next, we use this technique to learn the unitary transition operator in a recurrent neural network, and use this network to solve standard long-memory tasks.

Specifically, we define a general unitary RNN with recurrence relation

$$\mathbf{h}_t = f\left(\beta U \mathbf{h}_{t-1} + V\mathbf{x}_t + \mathbf{b}\right) \tag{3.47}$$

where f is a nonlinearity, β is a free scaling factor, U is our unitary matrix

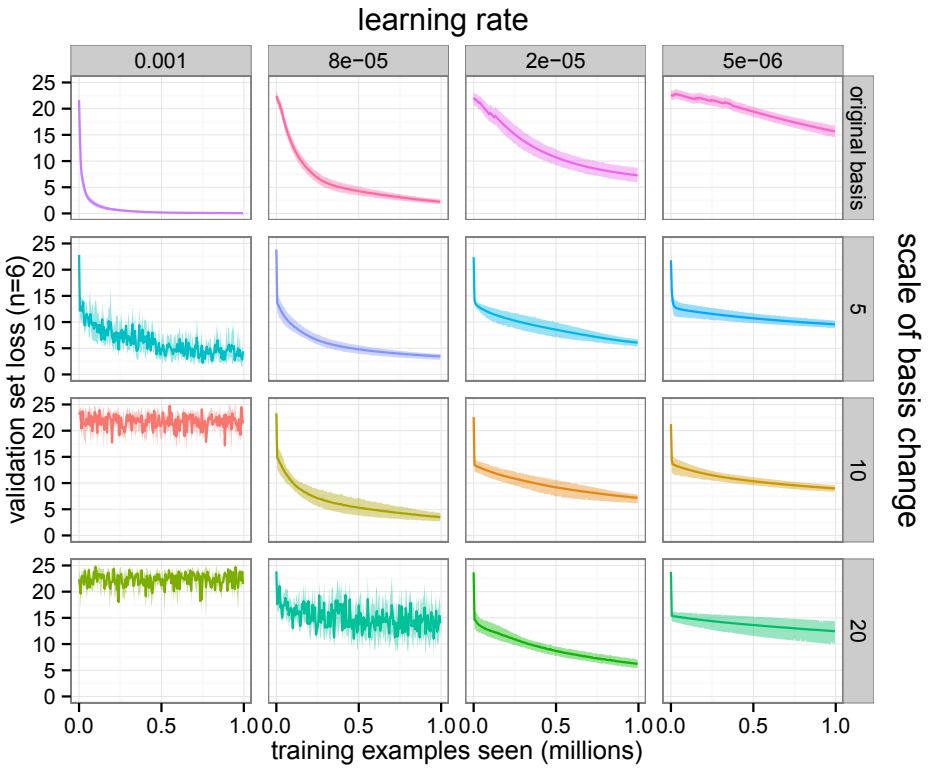


Figure 3.3: **Impact of changing the basis of the Lie algebra.** We consider the effects on learning of changing the basis (rows) and changing the learning rate (columns). For this experiment, $n = 6$. The first row uses the original basis. Other rows use change of basis matrices sampled from $\mathcal{U}(-c, c)$ where $c = \{5, 10, 20\}$. The learning rates decrease from the 'default' value of 0.001 used in the other experiments. Subsequent values are given by $\frac{0.001}{c^2}$ for the above values of c, in an attempt to rescale by the expected absolute value of components of the change of basis matrix. If the change of scale were solely responsible for the change in learning behaviour, we would therefore expect the graphs on the diagonal to look the same.

parametrised as in equation 3.15, $\mathbf{h}_t$ is the hidden state of the RNN and $\mathbf{x}_t$ is the input data at 'time-point' t. We refer to this as a 'general unitary RNN' (guRNN), to distinguish it from the restricted uRNN of Arjovsky et al. [6]. The β term was introduced specifically to try to compensate for the fact that U will never grow (or shrink) vector norms, but the nonlinearity f might. A β value above 1 therefore counteracts an overall shrinking effect introduced by f.

We use the guRNN on two standard tasks in the long-memory RNN literature: the 'adding problem' and the 'memory problem'. These were first described in Hochreiter and Schmidhuber [95], and are explored in sections 3.4.1 and 3.4.2. We compare our model (guRNN) with the restricted uRNN (ruRNN) parametrised as in equation 3.7, a LSTM [95], and the IRNN of Le et al. [131]. For the IRNN, we perform gradient clipping between $[-1, 1]$. The learning rate was set to $\alpha = 10^{-3}$ for all models except IRNN, which used $\alpha = 10^{-4}$. We used RMSProp [213] with decay 0.9 and no momentum. The batch size was 20.

Perhaps owing to our efficient gradient calculation (section 3.2.3) and simpler recurrence relation, our model runs faster than that of Arjovsky et al. [6] (in our implementation), by a factor of 4.8 and 2.6 in the adding and memory tasks respectively. This amounts to the guRNN processing 61.2 and 37.0 examples per second in the two tasks, on a GeForce GTX 1080 GPU.

3.4.1 Adding problem

In the adding problem, the RNN is given a 2-dimensional input sequence $\mathbf{x}$ of length T, where T is some large value. The first dimension consists of real values sampled uniformly from $[0, 1]$. The second dimension is binary, containing all zeroes except for two locations i and j. The index i is chosen uniformly from $[0, T/2]$, and j from $[T/2, T]$ to ensure the expected distance is $T/2$. These indices indicate which entries in the first dimension should be added to produce the result. The network therefore must learn to selectively record a particular input, store it until the second input is indicated, and produce their sum. An example is shown in Figure 3.4.

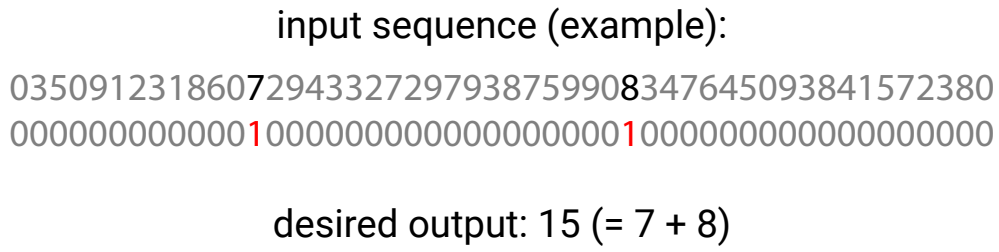


Figure 3.4: **An example of the adding problem.** The RNN receives a 2-dimensional input sequence of length T (large). The second, binary dimension of the input indicates which entries should be added (highlighted in red).Note that in practice, the first dimension contains real values in $[0, 1]$ and not integers as shown.

We show the comparison of different models in Figure 3.5. The performance is measured by mean squared error (MSE). The baseline model will always predict 1 (the expected value of the sum of the two uniformly-distributed values), which produces a baseline MSE of 0.167 - the variance of the distribution of the sum of two random variables X with $X \sim \mathcal{U}[0, 1]$. The LSTM ultimately outperforms other methods here, although Arjovsky [6] reports that for larger T (750), it begins to break down. This indicates that the LSTM may be optimal for shorter long sequences, while unitary approaches excel in the extreme long case. The approach proposed in this chapter, guRNN, performs similarly (albeit with instability) to the restricted uRNN of Arjovsky et al. [6] for $\beta = 1.4$ with a relu nonlinearity, and learns very slowly when $\beta = 1$. The identity-initialised RNN [131] exhibits large variance initially, but eventually approaches the performance of the other (successful) approaches. All models use a state size of $n = 30$, except for the restricted uRNN, which uses $n = 512$.

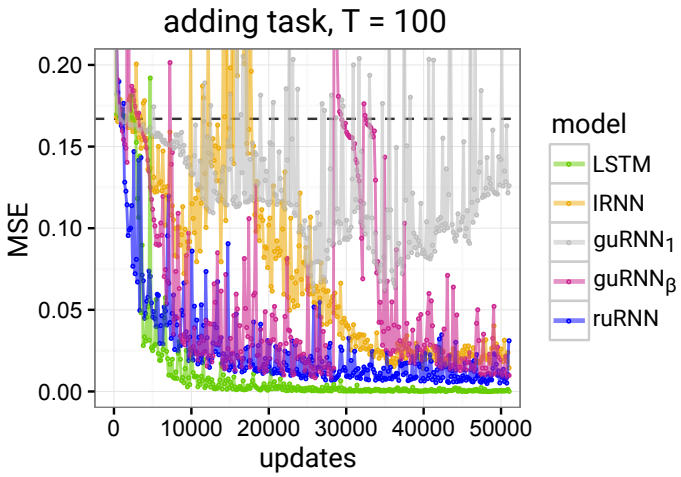


Figure 3.5: **Comparison of different long-memory RNN architectures on the adding problem.** Here $T = 100$. The scaling factor β in front of U in guRNN is 1.4 to compensate for the tendency of the nonlinearity (relu) to shrink gradients. The dotted line denotes the random baseline value of 0.167.

3.4.2 Memory problem

The memory problem requires the RNN to remember a sequence of length 10 over a time-span of T steps with no additional input. Then, after receiving a special token (shown as the number 9 in Figure 3.6), it must output the sequence it saw initially. This is depicted with an example in Figure 3.6.

input sequence (example):

1678453741000000000000000000000000000900000000000

desired output sequence:

000000000000000000000000000000000000001678453741

Figure 3.6: **An example of the memory problem.** The RNN receives a 1-dimensional input sequence of length T (large) + 20. The first 10 entries are of interest, and are followed by $T-1$ zeroes ('blank' token), then a single special token (shown as 9 in green), and 10 more zeroes. The special token indicates to the RNN that it should begin reproducing the memorised sequence. The RNN must then output $T + 10$ zeroes followed by the sequence of interest. Performance is measured by cross-entropy between the full true and output sequences.

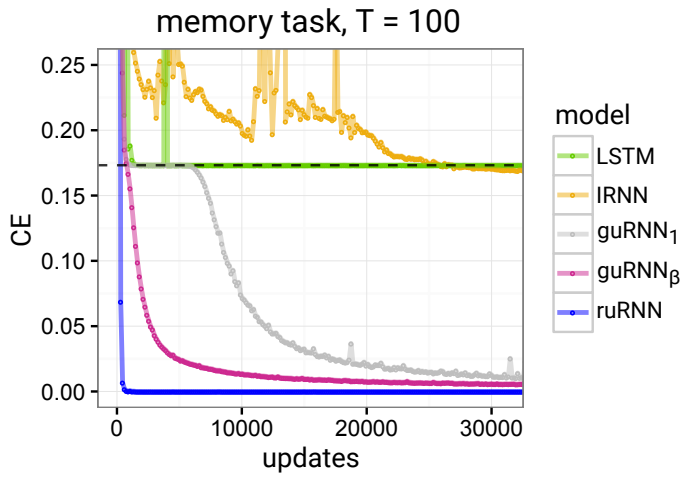


Figure 3.7: **Comparison of different long-memory RNN architectures on the memory problem.** Here $T = 100$. The dotted line denotes the random baseline value of $10\log(8)/(T + 20) = 0.173$.

In Figure 3.7 we show the relative performance, measured using categorical cross-entropy, of the different approaches considered. In this experiment, the restricted uRNN has a hidden state size of 128, as this was reported optimal in Arjovsky et al. [6]. All other models has a state size of 30. For the general uRNN (guRNN), we use a tanh nonlinearity, and $\beta = 1.05$ (in pink). We note that this optimal β is lower than in the adding task, which used a relu nonlinearity. This is not surprising, as relu discards more gradient information than tanh.

We see that the restricted uRNN [6] performs extremely well at this task, while the LSTM and IRNN [131] never outperform the baseline (dashed). This baseline is computed, following Arjovsky et al. [6], by imagining a 'memoryless' RNN which successfully outputs the blank token for $T + 10$ steps, and then outputs the sequence as 10 independently drawn random categories. With 8 categories available, the baseline average cross-entropy is therefore $10\log(8)/(T + 20)$.

3.5 Conclusions and future work

This chapter demonstrates how we can borrow machinery from Lie theory to enforce a hard constraint on the transition matrix in a recurrent neural network. More generally, we have exploited a parametrisation of unitary matrices in terms of their Lie algebra, which is importantly closed under addition, enabling this hard constraint to be satisfied during gradient-based optimisation. The use of such unitary matrices in recurrent neural networks is motivated by the need for models which can retain information over long input sequences, akin to those which arise in medical modelling.

The findings described in this chapter highlight further interesting questions:

Unitarity may not be enough Interestingly, although we have empirically shown that this parametrisation allows us to accurately learn unitary matrices (Section 3.3) when we apply it in a recurrent neural network (Section 3.4), other approaches such as the LSTM exhibit superior performance. This indicates that unitarity is not *sufficient* for success in these tasks. Interestingly, while we demonstrated that the Lie parametrisation can learn unitary matrices with higher accuracy than the restricted parametrisation of Arjovsky et al. [6] (Table 3.1), that approach demonstrates striking performance on the memory task (Figure 3.7). This suggests that that parametrisation contains an inductive bias which makes it particularly suited to that task. It may be that the parametrisation we propose is *too* general, and pointlessly gives access to parts of $U(n)$ which are not useful. Understanding to what extent this is the case is an interesting avenue of future research.

The role of the nonlinearity While our model guarantees unitarity of U, this is *not* sufficient to prevent gradients from vanishing. Recall Equation 3.4. Using a unitary matrix fixes $\|U\| = 1$, but beyond further restrictions (on V and $\mathbf{b}$) does nothing to control the norm of f', which is at *most* 1 for common nonlinearities. Designing a nonlinearity to better preserve gradient norms is a question for further research. Here, we simply scaled U by a constant multiplicative factor β to counteract the tendency of the nonlinearity to shrink gradients, denoting this setup by guRNN$_\beta$ in Figures 3.5 and 3.7 Confirming our intuition, this simple modification greatly improves performance on both tasks.

CHAPTER 4

SYNTHESISING MEDICAL DATA

> You could find out most things, if you knew the right questions to ask.
> Even if you didn't, you could still find out a lot.
>
> Iain M. Banks, *Player of Games*

Individual contributions *The work in this chapter was a collaboration between myself and Cristóbal Esteban, with supervision and guidance from Gunnar Rätsch. Cristóbal implemented and trained the conditional version of the model. I implemented the evaluation methods including MMD, and the heuristics for identifying model memorisation. I also implemented differentially private training. Everything else was done jointly.*

4.1 Why synthesise medical data?

One of the special challenges of working with medical data is that it is sensitive by nature, and often-times legally protected (e.g. HIPAA). For this reason, getting access to data often typically requires clinical collaborators associated with hospitals or other healthcare providers. Even if researchers have access to data, sharing and publishing that data is challenging. Even if it is legally permissible to publish anonymised data, achieving anonymisation in line with legal require-

ments is technically challenging, and as requirements and technology changes, patient privacy may still be compromised. A 2017 study [45] on allegedly-anonymised Australian health data reports successful re-identification using linkage attacks, a technique already demonstrated to re-identify individuals in the Netflix Prize dataset [164]. However, research groups publishing models developed on 'in-house' datasets results in a situation where many results can't be reproduced or even meaningfully compared to.

This poses two related issues for medical machine learners: a lack of reproducibility, and a lack of competition on shared tasks.

4.1.1 Reproducibility in medical machine learning

Lack of reproducibility in scientific research can come from many sources. In computational science, if the analysis environment can be exactly specified, including all hyperparameters, random seeds, library versions, and other implementation details (for example to the level of source code), and the original data can be provided, then results should be independently reproducible, assuming access to equivalent computational resources.

Lack of reproducibility therefore originates from:

1. Lack of documentation or transparency around analysis and computational environment
2. Lack of access to computational resources
3. Lack of access to data

In Henderson et al. [91], it is demonstrated in the context of deep reinforcement learning that different results can be obtained from otherwise identical experimental setups by simply varying the random seed. While this sensitivity to stochastic elements points towards a deeper issue in the fragility of the algorithms, it underpins the need to carefully document the experimental setup. They also highlight that differences in implementation can lead to differing results, emphasizing the need for analysis code to be made open source. Since medical research is often done in conjunction with companies or organisations who may block the open-sourcing of code (for business reasons), this can also be a barrier to reproducibility, but not one we address in this work.

Assuming researchers can publish all their analysis code and environment settings, it can still be practically impossible for other researchers to replicate results which required significant computational costs. For example, large-scale studies (such as Lucic et al. [139]) or especially high-performing architectures (such as OpenAI's Dota2-playing networks [172]) are challenging to independently verify. The barrier to reproducibility posed by computational resources applies more-so to recent results - assuming resources continue to drop in price, it becomes feasible for older results to be (potentially) replicated. Well-resourced institutions such as companies may publish non-replicable cutting edge work, but over time such results should be generally verifiable.

We address the challenge of data sharing in this work. Data accessibility is perhaps the hardest challenge, as good will and careful scientific practices cannot make up for inaccessible data. Even if work is fully documented and can be run on readily-accessible hardware, if the *inputs* to an analysis pipeline are unavailable, the outputs are meaningless. Data may not be available because

its exclusive access provides marginal value to an organisation (for example, data embargoes in genomics), or because there are logistical or legal barriers to sharing it. Regardless of the cause, we see this effect starkly in medical machine learning. In the last two years of the MLHC conference[1], 28 out of 55 papers (50.9%) used *only* 'in-house' or otherwise inaccessible data. Of the 27 papers using open-access datasets, one third used the same one - MIMIC-III[110]. Papers including any open-access (potentially with controls) dataset were considered to use accessible data. Interestingly, as shown in Johnson et al. [109], even within studies using the open-access dataset MIMIC-III, specifics of data preprepreprocessing such as the definition of exclusion criteria and endpoints resulted in a broad failure to reproduce mortality prediction results.

While scientific and medical insights can nonetheless be drawn from non-reproducible work, inability to compare is especially damaging for a field like machine learning where comparison can drive innovation. We explore this in the next section.

4.1.2 Benchmarks and competitions

The existence of publicly available datasets with pre-specified, well-defined tasks enables machine learning researchers to focus exclusively on optimising for performance. Data selection and task specification, while important skills in applied machine learning and data science, consume time which could otherwise be spent on developing new architectures, optimisation techniques, and other innovations to solve pre-specified problems. The existence of such benchmark tasks allows for a well-defined notion of 'state of the art' (SOTA), and

[1] https://www.mlforhc.org/

thus to the pursuit of breaking it. The broad appeal of this competitive machine learning philosophy is exemplified by the popularity of Kaggle[2] and the related ecosystem of courses, blog posts, and communities dedicated to pushing SOTA.

Within the academic community, the existence of SOTA (on various tasks) provides a lightning rod for research focus, enabling sub-communities to form around specified tasks, and provides a template for reporting results. This focus has not come without criticism. Langley [129] argues that benchmarks invariably prioritise tasks in which evaluation and experimentation is easier - classification and regression, rather than more complex tasks such as reasoning and problem-solving, and that this has lead to a narrowing of the field's focus. Wagstaff [224] calls for a shift towards 'real-world' problems, away from abstract metrics divorced from their practical impact, and proposes a set of concrete 'impact challenges'. However, even these criticisms arise because benchmarks are so *effective* at directing and driving research, to an extent which (it is argued) may be harmful. We cannot ignore their impact.

Perhaps the most famous benchmark is the MNIST hand-written digit database [132], which has been cited 14,726 times at the time of writing. Despite this dataset reaching its twentieth birthday this year, it remains a staple of machine learning, appearing not only in introductory texts and tutorials, but also research papers. Indeed, it appears in this very chapter. Before MNIST, the UCI Machine Learning Repository[51][3] was made available in 1987, and aimed to provide a set of standard (and diverse) tasks to the machine learning community, and persists to this day. More recently, and more practically interesting than MNIST is ImageNet[50][4] and its associated challenges. After the ImageNet

[2] `https://www.kaggle.com`
[3] `https://archive.ics.uci.edu/ml/index.php`
[4] `http://www.image-net.org`

dataset was published in 2009, the Large Scale Visual Recognition Challenges (ILSVRC) have been running annually, and error rates on the object detection tasks have fallen steadily. Moreover, new metrics have arisen - fast.ai recently claimed to have achieved 93% accuracy on ImageNet in 18 minutes at a cost of $40 [96], and Tencent report a training time of 4 minutes [107]. Results of these kind rely on innovation in the intersection of systems and machine learning, demonstrating that benchmarks need *not* narrow our focus - shared tasks can spawn innovation in unexpected directions.

Healthcare does not enjoy the same abundance of openly-available datasets with well-defined tasks, although some notable examples exist. As of 2000, the Computing in Cardiology conference, alongside PhysioNet hosts an annual challenge [73][5] based on some open clinical problem. The UCI repository [51] features (at the time of writing) 103 datasets classified as 'Life sciences'. In late 2017, the NIH published a deidentified dataset of chest xrays [227] labelled with thoracic disease categories (although the quality of the dataset has also been called into question [183]). In ophthalmology, the open-access Messidor dataset [47] contains 1200 eye fundus images, for the purpose of evaluating methods for diabetic retinopathy screening. Another similar dataset was published on Kaggle [75], however recent high-profile work in the automated identification of diabetic retinopathy [60] nonetheless uses a proprietary dataset. In critical care, two large open-access datasets exist - MIMIC-III [110], and the newer eICU collaborative research database [179]. Although these datasets don't come with specified tasks, a recent attempt has been made to define 'standard' tasks on MIMIC [85] including mortality prediction (long and short-term), forecasting length of stay, and phenotype classification. However, as noted in

[5] `https://physionet.org/challenge/`

Johnson et al. [109], even within a task as seemingly well-defined as mortality prediction, it can be challenging to reproduce even identical cohorts, due to non-specified processing and selection steps.

4.1.3 Synthetic data in machine learning and healthcare

Synthetic data is used widely across machine learning. Data can be generated according to easily-specified rules to generate a dataset with the desired properties for testing or training a machine learning system - for example, the adding and memorisation problems in the previous chapter. In time-series analysis for example, Krishnan et al. [125] generate data according to a nonlinear Gaussian state-space model. Lloyd and Ghahramani [138] uses synthetic data to demonstrate how maximum mean discrepancy can identify samples from different distributions (we revisit this in Section 4.4.1).

Owing to the challenge of sharing healthcare data, various attempts have already been made to generate usable synthetic medical data. Moniz et al. [161] generate synthetic EMRs designed to match statistics of a real EMR dataset such as seasonality, demographic distributions, and distributions of cases and diagnoses. The MIDAS project[6] makes synthetic epidemiological datasets available, for example using the SPEW package[7] for generating synthetic ecosystems. Synthea [226] is a software package to generate synthetic patient health histories using publicly available disease and demographic statistics. Despite these (and other) approaches to generating synthetic medical data, uptake in the machine learning community has been limited so far.

[6] https://www.epimodels.org/
[7] http://www.stat.cmu.edu/~spew/about/

4.2 Contributions

The work described in this chapter comprises three contributions:

1. An architecture (recurrent conditional generative adversarial network - RCGAN) for generating synthetic medical time-series
2. A method for evaluating the quality of the generated data using a supervised learning task
3. Analysis of the privacy implications of this model, through model memorisation, and an extension of RCGAN using differentially private training

Each of these points is further described in the following sections.

4.3 Generative adversarial networks

Generative adversarial networks (GANs) were proposed by Goodfellow et al. in 2014 and have inspired substantial research activity since. The core idea of a GAN is to train a generative model (the generator) using an objective provided by an adversarial model (the discriminator). The generator $G(\mathbf{z})$ maps points from a latent space $\mathbf{z} \in \mathcal{Z}$ to the data space $\mathbf{x} \in \mathcal{X}$, generating samples in $\mathcal{X}$. The discriminator $D(\mathbf{x})$ must distinguish between samples produced by the generator and those from some true data-generating process. The objective for the generator is then to produce convincing fakes - to maximise the error rate of the discriminator.

Concretely, the original GAN formulation is that G and D play a two-player

minimax game: (Goodfellow et al. [74] equation (1))

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D(G(\mathbf{z}))\right)\right] \tag{4.1}$$

In practice, optimisation means updating D and G in iterations using stochastic gradient descent.

The gradient update for the discriminator is

$$-\frac{\partial}{\partial\theta_D}\frac{1}{B}\sum_{i=1}^{B}\left[\log D\left(\mathbf{x}^i\right)\right] + \frac{1}{M}\sum_{i=1}^{M}\left[\log\left(1 - D\left(G\left(\mathbf{z}^i\right)\right)\right)\right] \tag{4.2}$$

where a batch of B data samples $\mathbf{x}$ are used, and a batch M of samples from the latent space. Typically, $B = M$.

The gradient update for the generator is

$$\frac{\partial}{\partial\theta_G}\frac{1}{M}\sum_{i=1}^{M}\log\left(1 - D\left(G\left(\mathbf{z}^i\right)\right)\right) \tag{4.3}$$

Here, the update only relies on M samples from the latent space - the generator never directly 'sees' samples from the true data distribution. The discriminator is responsible for directing the generator to produce 'realistic' samples.

If the discriminator is very poor - for example, if $D(\mathbf{x}) = 0.5$ for all $\mathbf{x}$, the gradients for the generator will go to zero, since D no longer depends on G. Equivalently, if the discriminator is very good - for example, $D(G(\mathbf{z})) = 0$ for all $\mathbf{z} \in \mathcal{Z}$ (or at least, in expectation over $p_{\mathbf{z}}$), then the gradients once again go to zero. Similarly, if the generator is very good, such that $G(\mathbf{z}) \sim p_{\text{data}}(\mathbf{x})$, the discriminator's objective reduces to

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x}] + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log\left(1 - D(\mathbf{x}^j)\right)\right] \tag{4.4}$$

which will push D towards $D(\mathbf{x}) = 0.5 \ \ \forall\mathbf{x}$, resulting in a static generator.

This means that training fails if the discriminator or the generator become mismatched. The relative number of training steps for the discriminator and generator can be considered a hyperparameter, and varying this ratio can aid with the mismatch problem. Further heuristics [189] have also been explored.

Other variants, such as Wasserstein GANs [81] and MMD-GANS [135] do away with the discriminator network, replacing its function with a distance measure such as the Wasserstein distance, or the maximum-mean discrepancy (see below), respectively. Variational autoencoders (VAEs) [121] are a competing class of generative model to GANs. VAEs have the desirable property of providing an explicit generative model from the latent space, as well as an encoder network, but empirically can provide less 'realistic' images than those from GANs. In this chapter we focus on GANs, but note that the use of recurrent VAEs for medical time series remains a question for further research.

4.3.1 Recurrent GANs

Much research into GANs and GAN variants has focused on producing realistic images. Images have two advantages:

1. Humans have highly adapted visual processing systems and can quickly recognise 'realistic' images - that is, evaluation is possible (if not rigorous) with manual inspection
2. We have sophisticated techniques for modelling and processing images, with a lot of collective experience in this domain

The second point is exemplified by the abundance of high-quality image GANs

exploiting CNNs, for example DCGAN architecture [181],

However, images are only one type of data. Researchers in natural language generation have also exploited GANs to synthesise text. Text is challenging because outputting discrete tokens requires introducing a non-differentiable process into the generator. This has been addressed by treating language generation as a reinforcement learning problem where outputting word tokens comprises a sequential decision-making problem [241] or using a feature-matching approach, effectively representing sentences using word embeddings [245], generating text using an LSTM in the typical way.

Synthetic medical data is somewhat similar to generating text as sequences of word embeddings - we need to create real-valued, multivariate time series. To do this, we construct a *recurrent* GAN (RGAN), where both generator and discriminator networks are recurrent neural networks. To produce a sequence of length T, the generator RNN receives T samples from the latent space. We found that in practice, it is sufficient to choose these latent samples independently. The discriminator then receives the input sequence and produces classifications at each time-point in the sequence. Since the correct label for a given sequence is either all 1 or all 0, this is a form of target replication.

Specifically, the discriminator is trained to minimise the average cross-entropy between its predictions *per time-step* and the labels of the sequence. If we denote by $\mathrm{RNN}(X)$ the vector or matrix comprising the T outputs from a RNN receiving a sequence of T vectors $\{\mathbf{x}_t\}_{t=1}^{T}$ ($\mathbf{x}_t \in \mathbb{R}^d$), and by $\mathrm{CE}(\mathbf{a}, \mathbf{b})$ the cross-entropy between sequences $\mathbf{a}$ and $\mathbf{b}$ averaged over time steps, then the discriminator loss for a pair $\{X_n, \mathbf{y}_n\}$ (with $X_n \in \mathbb{R}^{T \times d}$ and $\mathbf{y}_n \in \{1, 0\}^T$) is:

$$\mathrm{D}_{\text{loss}}(X_n, \mathbf{y}_n) = \mathrm{CE}(\mathrm{RNN}_{\mathrm{D}}(X_n), \mathbf{y}_n) \tag{4.5}$$

For real sequences, $\mathbf{y}_n$ is a vector of 1s, or 0s for synthetic sequences. In each training mini-batch, the discriminator sees both real and synthetic sequences.

The objective for the generator is then to 'trick' the discriminator into classifying its outputs as a true sequence, that is, it wishes to minimise the (average) cross-entropy between the discriminator's predictions on *generated* sequences and the 'true' label, the vector of 1s (we write as $\mathbf{1}$);

$$\mathrm{G}_{\text{loss}}(Z_n) = \mathrm{D}_{\text{loss}}(\mathrm{RNN}_{\mathrm{G}}(Z_n), \mathbf{1}) = \mathrm{CE}(\mathrm{RNN}_{\mathrm{D}}(\mathrm{RNN}_{\mathrm{G}}(Z_n)), \mathbf{1}) \tag{4.6}$$

Here Z_n is a sequence of T points $\{\mathbf{z}_t\}_{t=1}^{T}$ sampled *independently* from the latent/noise space $\mathbf{Z}$, thus $Z_n \in \mathbb{R}^{T \times m}$ since $\mathbf{Z} = \mathbb{R}^m$.

4.3.2 Conditional GANs

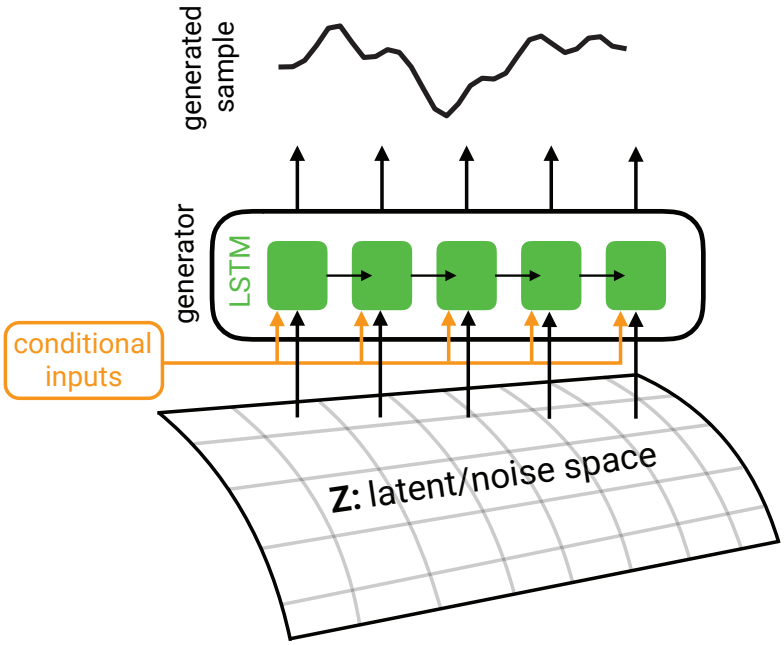
In the traditional GAN model, the inputs to the generator are samples drawn from the latent space according to some distribution $p_{(\mathbf{z})}$. These latent points serve as a source of stochasticity in the model, allowing the generator to act like a glorified transformation function in a random number generator, sampling points from p_{data}. To sample from a different distribution would in principle require retraining the GAN using data from this different distribution. However, sampling from different distributions would have many practical applications. In a medical setting for example, if we could generate samples of patients with specific conditions, we could use these in training environments. A creator using a generative model could choose to generate samples of specific objects, textures, sounds, etc. For creating synthetic training data, generating samples from multiple classes enables us to produce balanced training sets.

The idea of a *conditional* GAN is thus to produce a generator where the generated sample is *conditioned* on some input information. This takes the form of setting part of the latent point (which forms the input to the generator) to some value, and passing the same information to the discriminator. This was suggested in the original GAN paper [74], developed quickly after [158], and has since been elaborated upon. In Odena et al. [170], only the generator is fed the conditional (class) label, and the discriminator is required to both identify the data source (real or synthetic) and identify the class label. Isola et al. [104] condition the generator on more complex inputs, such as images and tags, to perform tasks like converting from thermal to colour photos, sketches to photos, and semantic labels to photos.

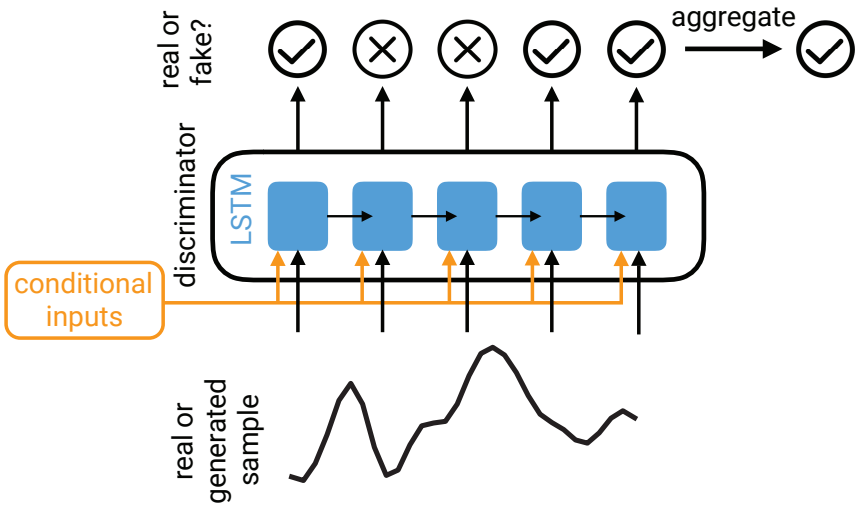
To form a conditional *recurrent* GAN, the inputs to each RNN are augmented with some conditional information $\mathbf{c}_n$ (for sample n) by concatenation at *each* time-step; $\mathbf{z}_{nt} \rightarrow [\mathbf{z}_{nt}; \mathbf{c}_n], \mathbf{x}_{nt} \rightarrow [\mathbf{x}_{nt}; \mathbf{c}_n]$. In this way the RNN cannot discount the conditional information through forgetting.

In this chapter, we use the CGAN to generate labelled training examples, where the label is used as $\mathbf{c}$. Generating such labelled data would also be possible by instead requiring the generator to output labels alongside its usual output. In this case, since the labels are binary values, it is simpler to use a conditional GAN approach. This also enables the specification of the desired label distribution in the generated synthetic data.

The schematics in Figure 4.1 show the proposed recurrent generator and recurrent discriminator respectively.



(a) **RCGAN generator architecture**. To generate a sequence of length T, the LSTM constituting the generator is given a sequence of T independent samples from $\mathcal{Z}$. The values of the synthetic sequence at time t are given by $\mathbf{x}_t = \tanh(W_o\mathbf{h}_t + \mathbf{b}_o)$. To produce samples outside the range $[-1, 1]$, a (learnable) scaling can be applied. The generator is trained by minimising the loss in Equation 4.6, alternating updates steps with the discriminator.



(b) **RCGAN discriminator architecture.** The discriminator classifies a sequence of length T as real or false by producing classifications at each time-point. The overall loss is then given by the average cross-entropy (log-loss) between its predictions and the true label, which is either a sequence of T zeroes, or T ones. This is shown in Equation 4.5. The discriminator is trained in alternating steps with the generator.

Figure 4.1: **Schematic of the RCGAN architecture.**

4.3.3 Generating realistic sequences with RCGAN

To demonstrate the models ability to generate 'realistic-looking' sequences in controlled environments, we consider several experiments on synthetic data. In the experiments that follow, unless otherwise specified, the synthetic data consists of sequences of length 30. We focus on the non-conditional model RGAN in this section.

Sine waves

The quality of generated sine waves are easily confirmed by visual inspection, but by varying the amplitudes and frequencies of the real data, we can create a dataset with nonlinear variations. We generate waves with frequencies in $[1.0, 5.0]$, amplitudes in $[0.1, 0.9]$, and random phases between $[-\pi, \pi]$. The left of Figure 4.2 shows examples of these signals, both real and generated (although they are hard to distinguish).

We found that, despite the absence of constraints to enforce semantics in the latent space (as in Chen et al. [36]), we could alter the frequency and phase of generated samples by varying the latent dimensions, although the representation was not 'disentangled', and one dimension of the latent space influenced multiple aspects of the signal.

Smooth functions

Sine waves are simple signals, easily reproduced by the model. In our ultimate medical application, we wish the model to reproduce complex physio-

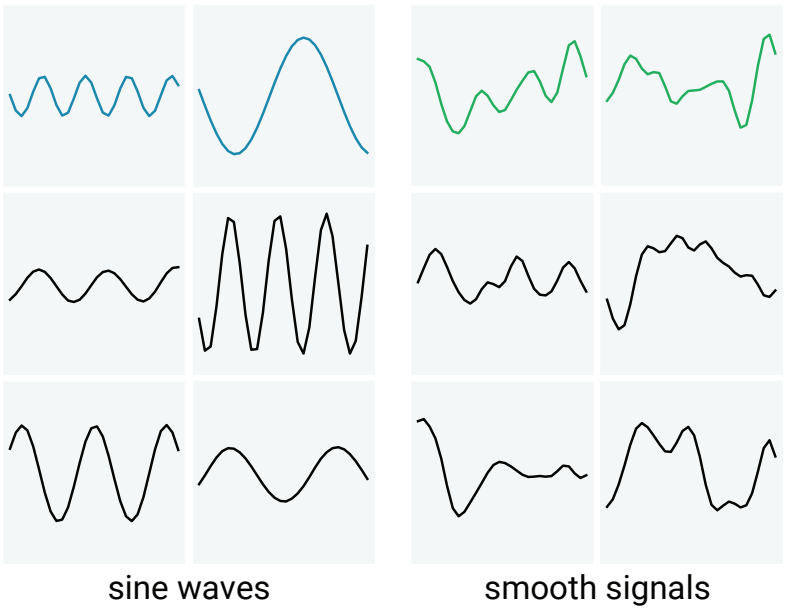


Figure 4.2: **Examples of real and generated samples of sine waves and smooth signals.** The top row shows real samples (in colour). The bottom two rows (in black) show generated samples. Smooth signals consist of samples from a Gaussian process with RBF kernel, evaluated at 30 equally-spaced points.

logical signals which may not follow simple dynamics. We therefore consider the harder task of learning arbitrary smooth signals. Gaussian processes offer a method to sample values of such smooth functions. We use a radial basis function (RBF) kernel to specify a GP with zero-valued mean function. We then draw 30 equally-spaced samples. This amounts to a single draw from a multivariate normal distribution with covariance function given by the RBF kernel evaluated on a grid of equally-spaced points. In doing so, we have specified exactly the probability distribution generated the true data, which enables us to evaluate generated samples under this distribution. The right of Figure 4.2 shows examples (real and generated) of this experiment. The main feature of the real and generated time series is that they exhibit smoothness with local correlations, and this is rapidly captured by the RGAN.

Because we have access to the data distribution, in Figure 4.4 we show how

the average (log) likelihood of a set of *generated* samples increases under the data distribution during training. This is an imperfect measure, as it is blind to the *diversity* of the generated samples - the oft-observed mode collapse, or 'Helvetica Scenario' [74] of GANs - hence we prefer the MMD^2 measure (see Figure 4.4 and Section 4.4.1). It is nonetheless encouraging to observe that, although the GAN objective is unaware of the underlying data distribution, the likelihood of the generated samples improves with training.

MNIST as a time series

The MNIST hand-written digit dataset is ubiquitous in machine learning research. Accuracy on MNIST digit classification is high enough to consider the problem 'solved', and generating MNIST digits seems an almost trivial task for traditional GANs. However, generating MNIST sequentially is less commonly done (notable examples are PixelRNN [218], and the serialisation of MNIST in the long-memory RNN literature [131]). To serialise MNIST, each 28×28 digit forms a 784-dimensional vector, which is a sequence we can aim to generate with the RGAN. This gives the added benefit of producing samples we can easily assess visually.

To make the task more tractable and to explore the RGAN's ability to generate *multivariate* sequences, we treat each 28x28 image as a sequence of 28, 28-dimensional outputs. We show two types of experiment with this dataset. In the first one, we train a RGAN to generate MNIST digits in this sequential manner. Figure 4.3 demonstrates how realistic the generated digits appear. In the second one, we use our proposed evaluation score, exploiting the fact that MNIST poses a classification task - this is explored in Sections 4.4.2 and 4.4.3.

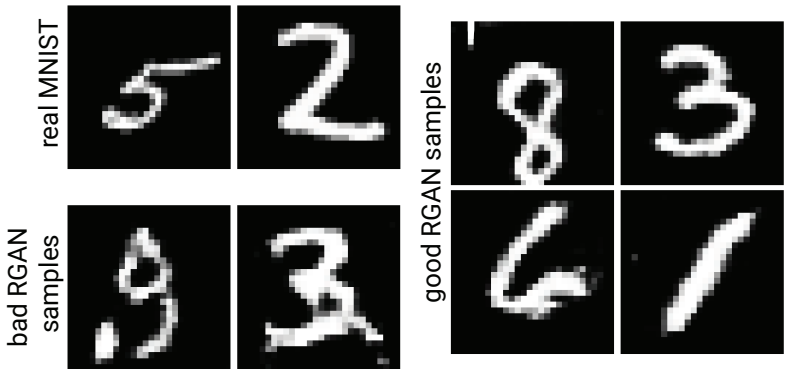


Figure 4.3: **Examples of real and generated MNIST samples.** Left top: real MNIST digits. Left bottom: unrealistic digits generated early in training. Right: digits with minimal distortion generated at epoch 100.

4.4 Evaluation of generative models

In the traditional Bayesian setting of generative models, we can use held-out data to estimate the predictive density of a model. Given a training dataset $\mathcal{D}$, if we can compute the posterior distribution over the parameters of the model θ, we can estimate the average density on a set of test samples $\{\mathbf{x}_i\}_{i=1}^N$ as follows:

$$\frac{1}{N}\sum_{i=1}^{N}\int p(\mathbf{x}_i|\theta)p(\theta|\mathcal{D})d\theta \tag{4.7}$$

In practice, the integral can be estimated by sampling from $p(\theta|\mathcal{D})$[219].

In the case of GANs, the above procedure is problematic because GANs are only generative in the sense of generating samples. A GAN with parameters θ defines an implicit model $p(\mathbf{x}|\theta)$ from which we can sample, but whose likelihood is not available. Parzen window estimates can be used to build this likelihood function[74], but Theis et al. [211] later demonstrated that artificially high likelihoods can be engineered using this approach. As demonstrated in Theis et al. [211], sample 'quality' (assessed visually) need not correspond to likelihood. Several methods have therefore been proposed to measure the performance of a GAN, which are reviewed briefly here. For a lengthier review, see Borji [28] and Xu et al. [237].

4.4.1 Existing evaluation methods for GANs

Manual inspection

A popular approach is to use manual inspection of generated samples - either visually assessing images, or reading sentences. Manual inspection is effective because 'sample quality' is intuitively understood to be 'realism', and human judges are adept at identifying real images, and text to a lesser extent. Moreover, while the intended use-case for many GANs is unclear, if the objective is to generate images or text perceived to be realistic by humans, then manual inspection should be the gold standard evaluation method.

Unfortunately, manual inspection is also problematic as a general-purpose evaluation method for several reasons:

1. It requires a human and therefore scales very poorly
2. Its use is limited in specialised domains. Judging the realism of natural images can be done by almost anyone[8], but identifying imperfections in specialist data (such as medical images or legal text) requires trained annotators. This increases the evaluation cost and makes it hard for non-experts to independently verify the performance of the model.
3. Human judgements are subjective and variable.

[8]Interestingly, even the data in CIFAR-10 [126] may already be domain-specific enough to warrant specialist annotators. In an evaluation using Amazon Mechanical Turk[189], the researchers achieved 95% accuracy at identifying generated samples, while the turkers achieved 78.7%.

Maximum mean discrepancy

We can take a 'black-box' approach to evaluating a generative model by simply asking if the samples it creates are from the same (or similar) distribution to the training (or test) data. However, as noted above, as GANs only give access to *samples* from the model, and inferred likelihoods can be deceptive, we need a method which can use sets of samples to compare distributions. This question - asking if two sets are samples are drawn from the same distribution, is precisely the setting of two-sample tests. In a line of work beginning in 2006 [26, 77], Gretton et *al.* demonstrated how a statistic called the maximum mean discrepancy (MMD) can be used to tackle the two-sample problem.

The MMD has an intuitive motivation. If we compute some *function* on each set of samples, if the *mean* value of that function differs between the sets (asymptotically), they cannot come from the same distribution. If the difference of the means is small, it indicates that the distributions are similar. Depending on the function and the underlying distributions, this difference in means can however vary (for example, if $f(x) = x$, then $\mathcal{N}(0, 1)$ appears identical to $\mathcal{N}(0, 2)$, but this would not be true for $f(x) = x^2$). To get around this, the MMD is defined as the supremum over all such mean differences, where the function is subject to some constraints. Specifically, if $X = \{x_1, \cdots, x_n\}$ are drawn i.i.d. from p and $Y = \{y_1, \cdots, y_n\}$ from q such that $x_i, y_j \in \mathcal{X}$, and $f : \mathcal{X} \to \mathbb{R}$ where $f \in \mathcal{F}$, then the MMD is defined as

$$\text{MMD}[\mathcal{F}, p, q] = \sum_{f \in \mathcal{F}} (\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]) \tag{4.8}$$

where $\mathcal{F}$ is a special class of functions described now.

The challenge of choosing $\mathcal{F}$ is that it should be rich enough to guarantee

that

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{y\sim q}[f(y)] \leftrightarrow p = q \tag{4.9}$$

but not so complex that finite-sample evaluations are uselessly inaccurate.

The approach taken in Gretton et al. [77] is to define $\mathcal{F}$ to be the unit ball in a reproducing kernel Hilbert space (RKHS). In this case, $\mathcal{F}$ satisfies the requirement of Equation 4.9 because the resulting MMD statistic is consistent (it has a type-II error rate of zero in the limit of large sample size) [78], and it satisfies the criterion of usefulness in the finite-sample regime because the type-II error converges to zero at a rate of $\mathcal{O}(n^{1/2})$, where n is the sample size. Moreover, the statistic of interest (MMD^2) can be computed in $\mathcal{O}(n^2)$.

What then is the MMD in the RKHS? Moving to consider MMD^2 (because this is equal to the squared distance between *mean* embeddings of the distributions p and q in the RKHS [26]), we have

$$\text{MMD}^2[\mathcal{F}, p, q] = \mathbb{E}_{x,x'}[k(x, x')] - 2\mathbb{E}_{x,y}[k(x, y)] + \mathbb{E}_{y,y'}[k(y, y')] \tag{4.10}$$

where k is the unique reproducing kernel associated with the RKHS in question.

An unbiased estimate of MMD^2 can be obtained as [78];

$$\begin{aligned}\widehat{\text{MMD}}_u^2 =& \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j\neq i}^{n} k(x_i, x_j) - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j) \\ &+ \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j\neq i}^{m} k(y_i, y_j)\end{aligned} \tag{4.11}$$

Thus, the computational cost is $\mathcal{O}((n+m)^2)$.

The choice of k (and thus the RKHS) has important implications for the MMD statistic. Importantly, the RKHS needs to be *universal* - this implies the MMD is a metric [77], which gives Equation 4.9. While universality of kernels

is beyond the scope of this work (see Sriperumbudur et al. [200] for a discussion), it has been shown [203] that the RKHSs associated with the Gaussian and Laplace kernels are universal. Therefore, we consider Gaussian (RBF) k in this work, acknowledging that defining an appropriate kernel between multivariate, likely non-stationary time-series is an area of active research [35, 46]. We select the bandwidth σ of the RBF kernel as per Sutherland et al. [208] to maximise the estimate of the t-statistic of the power of the MMD test:

$$\hat{t} = \frac{\widehat{\mathrm{MMD}}^2}{\sqrt{\hat{V}}} \tag{4.12}$$

where V is the asymptotic variance of the estimator of MMD^2. This is performed on a held-out set. Moreover, to address the challenge of defining an appropriate kernel on time series, we follow Sutherland et al. [208] and use sum of Gaussian kernels with different bandwidths, which are optimised simultaneously as above.

While MMD is described here as an evaluation measure, it can also be used as the objective in training a generative model. This is the approach taken in Li et al. [135] and examined further in Bińkowski et al. [17].

In this chapter, we use MMD as one method to evaluate samples from the RCGAN.

We describe other popular evaluation methods (Inception score, FID, and KID) below, but note that as these methods are all specific to image GANs as they rely on a pre-trained object detector (the Inception network [209]). Therefore, without modification they are not appropriate for use on time-series data.

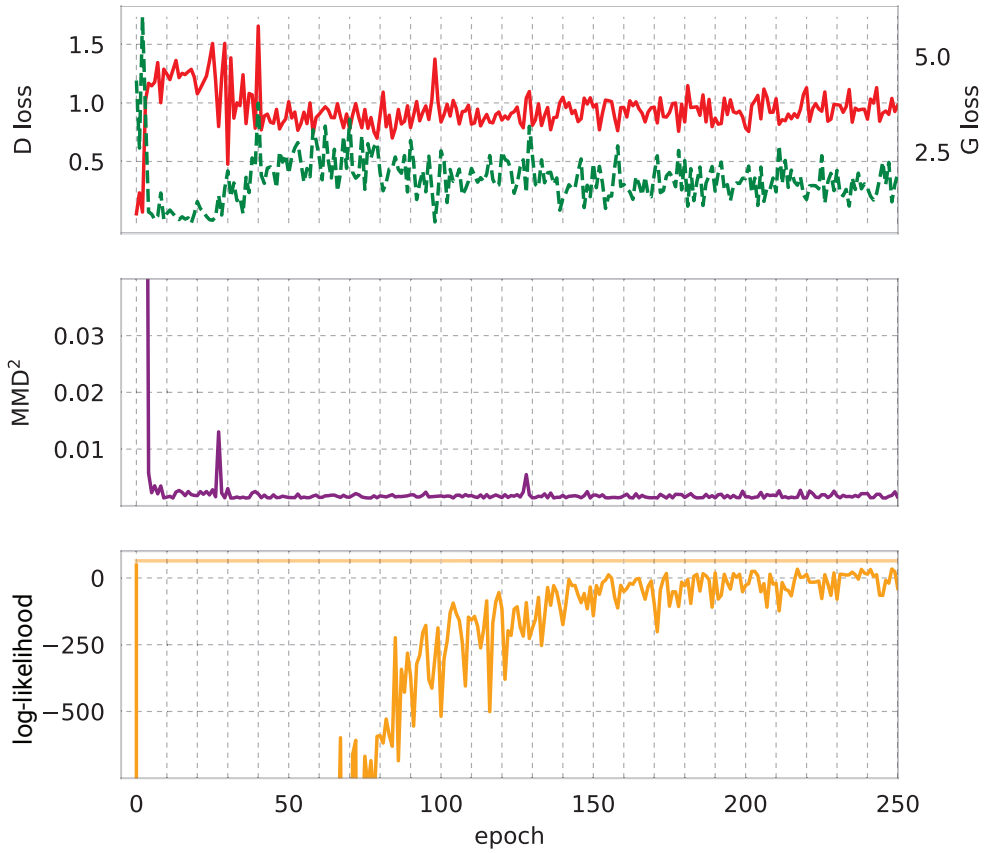


Figure 4.4: **Learning curves, MMD score, and held-out log-likelihood for RGAN generating smooth sequences.** Trace of generator (dotted), discriminator (solid) loss, MMD^2 score and log likelihood of generated samples under the data distribution during training for RGAN generating smooth sequences (output in Figure 4.2.)

Inception Score

The idea behind the inception score (IS) [189] is to automate the evaluation process by replacing a human's judgement of the image with that of a pre-trained image classifier, specifically the Inception model [209]. The Inception model is a 42-layer neural network trained to achieve state of the art (in CVPR 2016) on classification task in the ImageNet Large-Scale Visual Recognition Challenge

(ILSVRC) 2012 [187]. ILSVRC 2012 consists of images containing objects from 1000 categories, with 1.2 million training examples and 150,000 test/validation examples. The images are from the ImageNet [50] dataset, which was constructed by downloading images from Google image searches using terms from WordNet [155]. The categories include, for example, *border collie*, *Siberian husky* (12% of the classes are breeds of dog), *pool table*, *sundial*. The inception score is then based on the belief that high-quality data will contain *specific identifiable* objects, but also a full *diversity* of objects, across many samples. Using the trained Inception model $\mathcal{M}$, the conditional label probabilities for an input image $\mathbf{x}$ can be computed, $\mathcal{M}(\mathbf{x}) = p(y|\mathbf{x})$. If $\mathbf{x}$ contains a meaningful object (that $\mathcal{M}$ can recognise), these probabilities should have low entropy. At the same time, the *marginal* distribution, $p(y)$ have high entropy, such that all categories are represented. The marginal is calculated as $p(y) = \int d\mathbf{x} p(y|\mathbf{x})p(\mathbf{x})$, where $\mathbf{x} = G(z)$ where G is the deterministic generator, and z is a random variable with some distribution, usually Gaussian. In practice this is computed by averaging over a large number (they recommend 50,000) of samples from the generator. To combine the desire for $p(y)$ to be high entropy, and $p(y|\mathbf{x})$ (for all $\mathbf{x}$) to be low entropy, they formulate the score as follows:

$$IS = \exp\left(\mathbb{E}_{\mathbf{x}}\left[KL(p(y|\mathbf{x})|p(y))\right]\right) \tag{4.13}$$

Although it has seen widespread use [98, 147, 231], the inception score has several prominent limitations:

- It is not defined for datatypes which cannot be meaningfully passed through the Inception network, limiting it to GANs which generate images. Even within image-generating GANs, if the images do not contain objects from the ImageNet category the score will clearly fail. In Rosca

et al. [186] this is addressed by training an Inception-style net on CIFAR-10 to evaluate a CIFAR-10-trained generative model[9].

- As highlighted by Barratt and Sharma [11], since the inception score uses a complex neural network, small differences in implementation can influence the score. They found that the score can be 11.5% higher (on CIFAR images) using a Keras [40] implementation, compared to Torch [43].
- The true data distribution is not used anywhere. If a GAN trained on MNIST were somehow to produce perfect examples from ImageNet, it would achieve high inception score while producing data completely different to its training distribution.

The mode score (Che et al. [33], corrected in Xu et al. [237]) attempts to address the final point by including an additional KL term between the marginal distribution based on the *real* data and the generated data. Intuitively, this tries to control for the distribution of ImageNet classes in the training data.

FID

Another inception-based score is the Fréchet Inception Distance (FID) [92], which uses the Inception model as a feature extractor rather than a classifier. Taking the activations in an intermediate layer of the Inception model as the representation for a sample, FID models these representations as samples from a Gaussian distribution whose moments can be empirically estimated. Applying the same procedure to the real and generated data, FID then computes the

[9]However, Barratt and Sharma [11] observes that the practice nonetheless persists of applying the inception score to inappropriate datasets such as CIFAR-10, which has only 10 object classes, as well as celebrity faces [8].

Fr'echet distance (equivalently Wasserstein-2 distance) between the Gaussian distributions fit to the two sets of representations. By Dowson and Landau [53], this distance has an analytic expression, producing

$$FID(\mathbb{P}_r, \mathbb{P}_g) = \|\mu_r - \mu_g\| + \text{Tr}\left(\mathbf{C}_r + \mathbf{C}_g - 2\left(\mathbf{C}_r\mathbf{C}_g\right)^{1/2}\right) \tag{4.14}$$

where $\mathbb{P}_i = \mathcal{N}(\mu_i, \mathbf{C}_i)$.

Kernel Inception Distance

Similar but distinct to the Fréchet Inception Distance is the *Kernel* Inception Distance (KID). KID uses a representation of the data derived from an Inception model, but unlike FID which then computes a Fréchet distance, KID computes the MMD with a polynomial kernel between samples under the Inception-derived representation. This is equivalent to calculating MMD on the samples using a kernel which first passes its inputs through the Inception representation. The advantage of KID over FID is that the Gaussian assumption on the distribution of the representations is lifted, and as shown by Bińkowski et al. [17], has an unbiased estimator.

4.4.2 Our task-based score

While MMD (Section 4.4.1) provides a measure of sample quality, uncertainty about the appropriateness of the kernel choice leads us to devise another evaluation strategy. Since our generative model has an intended use-case (synthesising realistic data which could potentially be used to train models), we can use this to motivate an evaluation strategy. Specifically, we propose a novel (at

the time of the work [103]) method for evaluating the output of a GAN when a supervised task can be defined on the domain of the training data. We call it '**T**rain on **S**ynthetic, **T**est on **R**eal' (TSTR).

Simply put, we use a dataset generated by the GAN to train a model, which is then tested on a held-out set of true examples. This requires the generated data to have labels - we can either provide these to a conditional GAN, or use a standard GAN to generate them in addition to the data features. In this work we opt for the former (as in RCGAN, Figure 4.1). We present the pseudo-code for this GAN evaluation strategy in Algorithm 1.

Recently, a modification of TSTR has been proposed [112] which exploits multiple tasks and classifiers to produce a more robust evaluation for the purpose of comparing model performance.

Algorithm 1 (TSTR) Train on Synthetic, Test on Real

```
train, test = split(data)
discriminator, generator = train_GAN(train)
with labels from train:
    synthetic = generator.generate_synthetic(labels)
    classifier = train_classifier(synthetic, labels)
    If validation set available, optionally optimise GAN
    over classifier performance.
with labels and features from test:
    predictions = classifier.predict(features)
    TSTR_score = score(predictions, labels)
```

Train on Real, Test on Synthetic (TRTS) Similar to the TSTR method proposed above, we can consider the reverse case, called 'Train on Real, Test on Synthetic' (T**R**T**S**). In this approach, we use real data to train a supervised model on a set of tasks. Then, we use the RCGAN to generate a synthetic test set for evaluation. In the case (as for MNIST) where the true classifier achieves high

accuracy, this serves to act as an evaluation of the RCGAN's ability to generate convincing examples of the labels, and that the features it generates are realistic. Unlike the TSTR setting however, if the GAN suffers mode collapse, TRTS performance will not degrade accordingly, so we consider TSTR the more interesting evaluation.

4.4.3 Performance of RCGAN with TSTR

Although we have shown samples from RCGAN in section 4.3.3, as this section has highlighted, visual assessment of GAN samples is a problematic evaluation method. We therefore use TSTR to evaluate the RCGAN-generated samples in this section.

MNIST

For the second experiment, we downsample the MNIST digits to 14x14 pixels, and consider the first three digits (0, 1, and 2). With this data we train a RCGAN and subsequently perform the TSTR (and TRTS) evaluations explained above, for the task of classifying the digits. That is, for the TSTR evaluation, we generate a synthetic dataset using the GAN, using the real training labels as input. We then train a classifier (a convolutional neural network) on this data, and evaluate its performance on the real held-out test set. Conversely, for TRTS we train a classifier on the real data, and evaluate it on a synthetic test dataset generated by the GAN. Results of this experiment are show in Table 4.1. To obtain error bars on the accuracies reported, we trained the RCGAN five times with different random initialisations. The TSTR result shows that the RCGAN generates

Table 4.1: **TSTR results for MNIST.** Scores obtained by a convolutional neural network when: a) trained and tested on real data, b) trained on real and tested on synthetic, and c) trained on synthetic and tested on real data.

	Accuracy
Real	0.997 ± 0.00008
TRTS	0.976 ± 0.0013
TSTR	0.951 ± 0.0007

synthetic datasets realistic enough to train a classifier which then achieves high performance on real test data. The TRTS result shows that the synthetic examples in the test set match their labels to a high degree, given the accuracy of the classifier trained on real data is very high.

eICU

One of the main goals of this chapter is to build a model capable of generating realistic medical datasets, and specifically ICU data. For this purpose, we based our work on the recently-released Philips eICU database[10]. This dataset was collected by the critical care telehealth program provided by Philips. It contains around 200,000 patients from 208 care units across the US, with a total of 224,026,866 entries divided in 33 tables.

From this data, we focus on generating the four most frequently recorded, regularly-sampled variables measured by bedside monitors: oxygen saturation measured by pulse oximeter (SpO2), heart rate (HR), respiratory rate (RR) and mean arterial pressure (MAP). In the eICU dataset, these variables are measured every five minutes. To reduce the length of the sequences we consider, we downsample to one measurement every fifteen minutes, taking the median value in each window. This greatly speeds up the training of our LSTM-based

[10] `https://eicu-crd.mit.edu/`

GAN while still capturing the relevant dynamics of the data.

In the following experiments, we consider the *beginning* of the patient's stay in the ICU, considering this a critical time in their care. We focus on the first 4 hours of their stay, which results in 16 measurements of each variable. While medical data is typically fraught with missing values, in this work we circumvented the issue by discarding patients with missing data (after downsampling). After preprocessing the data this way, we end up with a cohort of 17,693 patients. Most restrictive was the requirement for non-missing MAP values, as these measurements are taken invasively.

To perform the TSTR evaluation, we need a supervised task (or tasks) on the data. A relevant question in the ICU is whether or not a patient will become 'critical' in the near future (indeed, this is the focus of Chapter 4) - a kind of early warning system. For a model generating dynamic time-series data, this is especially appropriate, as *trends* in the data are likely most predictive. Based on our four variables (SpO2, HR, RR, MAP) we define 'critical thresholds' and generate binary labels of whether or not that variable will exceed the threshold in the next hour of the patient's stay - that is, between hour 4 and 5, since we consider the first four hours 'observed'. The thresholds are shown in the columns of Table 4.2. There is no upper threshold for SpO2, as it is a percentage with 100% denoting ideal conditions.

As for MNIST, we 'sample' labels by drawing them from the real data labels, and use these as conditioning inputs for the RCGAN. This ensures the label distribution in the synthetic dataset and the real dataset is the same, respecting the fact that the labels are not independent (a patient is unlikely to simultaneously suffer from high and low blood pressure), while not leaking personal health

Table 4.2: **TSTR results on eICU.** Performance of random forest classifier for eICU tasks when trained with real data and when trained with synthetic data (test set is real), including random prediction baselines. AUPRC stands for area under the precision-recall curve, and AUROC stands for area under ROC curve. Italics denotes those tasks whose performance were optimised in cross-validation. SpO2 = oxygen saturation measured by pulse oximetry; HR = heart rate; RR = respiratory rate; MAP = mean arterial pressure.

		SpO2 < 95	HR < 70	*HR > 100*
AUROC	real	0.9587 ± 0.0004	0.9908 ± 0.0005	0.9919 ± 0.0002
	TSTR	0.88 ± 0.01	0.96 ± 0.01	0.95 ± 0.01
	random	0.5	0.5	0.5
AUPRC	real	0.9059 ± 0.0005	0.9855 ± 0.0002	0.9778 ± 0.0002
	TSTR	0.66 ± 0.02	0.90 ± 0.02	0.84 ± 0.03
	random	0.16	0.26	0.18

		RR < 13	RR > 20	MAP < 70	MAP > 110
AUROC	real	0.9735 ± 0.0001	0.963 ± 0.001	0.9717 ± 0.0001	0.960 ± 0.001
	TSTR	0.86 ± 0.01	0.84 ± 0.02	0.875 ± 0.007	0.87 ± 0.04
	random	0.5	0.5	0.5	0.5
AUPRC	real	0.9557 ± 0.0002	0.891 ± 0.001	0.9653 ± 0.0001	0.8629 ± 0.0007
	TSTR	0.73 ± 0.02	0.50 ± 0.06	0.82 ± 0.02	0.42 ± 0.07
	random	0.26	0.1	0.39	0.05

information.

Following Algorithm 1, we train the RCGAN for 1000 epochs, saving one version of the dataset every 50 epochs. Afterwards, we evaluate the synthetic data using TSTR. We use cross validation to select the best synthetic dataset based on the classifier performance, but since we assume that it might be also used for unknown tasks, we use only 3 of the 7 tasks of interest to perform this cross validation step (denoted in italics in Table 4.2). The results of this experiment are presented in Table 4.2, which compares the performance achieved by a random forest classifier that has been trained to predict the 7 tasks of interest, in one experiment with real data and in a different experiment with the synthetically generated data.

4.5 Privacy implications

Since medical data contains sensitive personal information, protecting patient privacy is one of the reasons these datasets cannot be shared easily. For a synthetic dataset to be viable, it is necessary to ensure that it does not leak too much information about the training data. However, machine learning systems have already been shown to be vulnerable to various attacks on privacy. In Fredrikson et al. [65], a model inversion attack is described to extract examples from the training data of a facial recognition system, given the adversary knows one of the output labels of the system. This requires limited side information. Shokri et al. [194] describe a membership attack, where they infer if a given (known) record was used to train a model, given black-box access to the model. Recently, similar attacks have been described against GANs [86]. These approaches generally rely on the fact that neural networks can 'memorise' training data. In this section we explore model memorisation in GANs, and go on to describe how we can use differentially private training to produce a privacy-preserving generative model.

4.5.1 Model memorisation

One of the primary failure modes of statistical models is that of *overfitting*, or failing to generalise. In a supervised learning setting, this can obviously happen when the model becomes sensitive to spurious correlations between noise features and labels. In this sense, the model has encoded aspects of the data which are, in truth, unnecessary for the task. We can think of this as a form of memorisation of the training data.

In the context of generative models, *generalisation* is not well-defined. We have neither a notion of model performance (notwithstanding the explorations in section 4.4.2), nor of this performance on a test set. The generator network simply produces examples. However, generative models such as GANs are known to be at risk of a phenomenon dubbed 'mode collapse', which resembles overfitting. Intuitively, mode collapse occurs when the generator produces only a small number of examples, typically closely resembling examples from the training set. Given the naive GAN objectives in Goodfellow et al. [74], we can see this can be a winning strategy: if G maps all $\mathbf{z}$ to the *same* $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ happens to be in the training set, the generator is 'successful', in that the discriminator can't achieve perfect performance - it must either identify $\hat{\mathbf{x}}$ as real or fake, and it will always be wrong some of the time. This is especially problematic if there are many examples in the training data which look like $\hat{\mathbf{x}}$ - if there is a *mode* in the data around $\hat{\mathbf{x}}$, it can be sufficient for G to output examples similar to $\hat{\mathbf{x}}$ and confuse the discriminator. Approaches to address or prevent mode collapse include asking the discriminator to look at many examples simultaneously (minibatch discrimination or averaging [39, 189]), introducing auxiliary networks [201], and modifications to the loss function as in Metz et al. [148].

To identify memorisation, we perform three tests - one qualitative, two statistical, outlined in the following subsections. In the statistical tests, the notion of memorisation we consider is the idea that the GAN would *more likely* produce samples that *more closely* resemble training set samples than unseen samples from the same distribution (that is, test set samples). In the absence of a more rigorous definition of memorisation in GANs, this notion is intuitively appealing, highlights the potential privacy risk of memorisation, and produces testable hypotheses, as we show below.

Analysis: Comparing the distribution of reconstruction errors

To test if the generated samples look too similar to the training set, we could generate a large number of samples and report the distance to the nearest neighbour (in the training set) to each generated sample. We could compare the distribution of these distances with those comparing the generated samples and a held-out test set. However, to get an accurate estimate of the distances, we may need to generate many samples, and correspondingly calculate many pairwise distances. Instead, we *intentionally generate* the nearest neighbour to each training (or test) set point, and then compare the distances.

We generate these nearest neighbours by minimising the reconstruction error between target y and the generated point;

$$\mathcal{L}_{\text{recon}(y)}(Z) = 1 - K(G(Z), y)$$

where K an RBF kernel (similar to that in Section 4.4.1, with bandwidth σ chosen using the median heuristic [29]). We find Z by minimising the error until approximate convergence (when the gradient norm drops below a threshold).

We can then ask if we can distinguish the *distribution* of reconstruction errors for different input data. Specifically, we ask if we can distinguish the distribution of errors between the training set and the test set. The intuition is that if the model has "memorised" training data, it will achieve identifiably lower reconstruction errors than with the test set. We use the Kolmogorov-Smirnov two-sample test to test if these distributions differ. For the RGAN generating sine waves, the p-value is 0.2 ± 0.1, for smooth signals it is 0.09 ± 0.04, and for the MNIST experiment shown in Figure 4.3 it is 0.38 ± 0.06. For the MNIST trained with RCGAN (TSTR results in Table 4.1), the p-value is 0.57 ± 0.18. We

conclude that the distribution of reconstruction errors is not significantly different between training and test sets in any of these cases, indicating that the nearest neighbours to the training set are not significantly closer or farther than the nearest neighbours to the test set.

Analysis: Comparing the generated samples

Rather than using a nearest-neighbours approach (as in Section 4.5.1), we can use the MMD three-sample test [29] to compare the full set of generated samples. With X being the generated samples, Y and Z being the test and training set respectively, we ask if the MMD between X and Y is less than the MMD between X and Z. The test is constructed in this way because we expect that if the model *has* memorised the training data, that the MMD between the synthetic data and the training data will be significantly lower than the MMD between the synthetic data and test data. In this case, the hypothesis that MMD(synthetic, test) $\leq$ MMD(synthetic, train) will be false. We are therefore testing (as in Section 4.5.1) if our null hypothesis (that the model has *not* memorised the training data) can be rejected.

The average p-values we observed were: for the eICU data in Section 4.4.3: 0.40 ± 0.05, for MNIST data in Section 4.4.3: 0.47 ± 0.16, for sine waves: 0.41 ± 0.07, for smooth signals: 0.07 ± 0.04, and for the higher-resolution MNIST RGAN experiments in Section 4.3.3: 0.59 ± 0.12 (before correction for multiple hypothesis testing). We conclude that we cannot reject the null hypothesis that the MMD between the synthetic set and test set is at most as large as the MMD between the synthetic set and training set, indicating that the synthetic samples do not look more similar to the training set than they do to the test set.

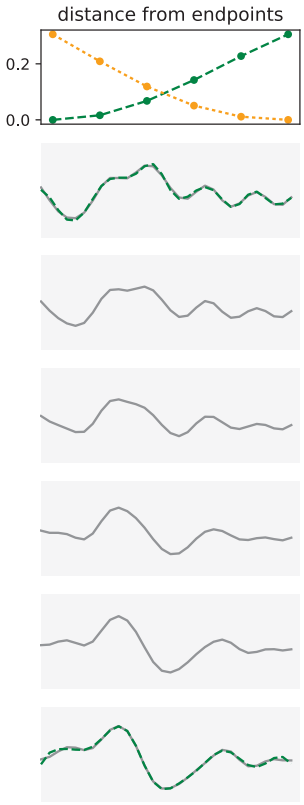


Figure 4.5: **Assessing model memorisation using interpolation in latent space.** Here, we take two training examples (bottom and second-from-top plots, dashed green lines) and back-project them to find their closest points in latent space. We then linearly interpolate these points and pass them through the generator, producing intermediate samples. These are shown in grey. The top graph shows the distance in sample space (using the RBF kernel) between the intermediate generated sample and both of the endpoints (respectively in green and orange). If model memorisation had occurred, we would expect the model to preferentially generate samples that look very similar to either of the endpoints, switching between these options perhaps half-way through. Instead, the generator produces samples which appear to vary smoothly as the latent point is varied.

Analysis: Interpolation

Suppose that the model has over-fit, so the implicit distribution is highly peaked in near training examples, and most points in latent space map to (or near) training examples. If we take a smooth path in the latent space, we expect that

at each point, the corresponding generated sample will have the appearance of the 'closest' (in latent space) training example, with little variation until we reach the attractor basin of another training example, at which point the samples switch appearance.

We test this qualitatively as follows: sample a pair of training examples (confirming by eye that they don't look too similar), and back-project them into the latent space to find the closest corresponding latent point, as described above. We then linearly interpolate between those latent points, producing samples from the generator at each point. Figure 4.5 shows an example using the 'smooth function' dataset. The samples show a clear incremental variation between start and input sequences, contrary to what we would expect if the model had memorised the data.

4.5.2 Differentially private GAN training

In Section 4.5.1 we perform empirical analyses of the data generated by the GAN, to test if it has memorised the training data. While we do not observe evidence to support memorisation (and thus privacy violation), medical data is highly sensitive and guaranteed privacy is often required. Therefore, we investigated the use of a differentially private training procedure for the GAN.

Differential privacy

Differential privacy [54] is concerned with the influence of the presence or absence of individual records in a database. Intuitively, differential privacy places

bounds on the probability of obtaining the same result (in our case, an instance of a trained GAN) given a small perturbation to the underlying dataset. If the training procedure guarantees (ϵ, δ) differential privacy, then given two 'adjacent' datasets (differing in one record) D, D',

$$P[\mathcal{M}(D) \in S] \leq e^{\epsilon} P[\mathcal{M}(D') \in S] + \delta \tag{4.15}$$

where $\mathcal{M}(D)$ is the GAN obtained from training on D, S is any subset of possible outputs of the training procedure (any subset of possible GANs), and the probability P takes into account the randomness in the procedure $\mathcal{M}(D)$. Thus, differential privacy requires that the distribution over GANs produced by $\mathcal{M}$ must vary 'slowly' as D varies, where ϵ and δ bound this 'slowness'.

Differentially private SGD

Inspired by a recent preprint [13], we apply the differential private stochastic gradient descent (DP-SGD) algorithm of Abadi et al. [1] to the discriminator, as the generator does not see the private data directly.

The DP-SGD differs from traditional SGD in the addition of two steps[11]

1. Per-example gradients are clipped to have a maximum norm of CB, where B is the batch size and C is a sensitivity parameter.
2. After averaging clipped gradients, isotropic Gaussian noise is added with standard deviation $C\sigma$, where σ is another parameter of the algorithm.

The effect of clipping is to bound the sensitivity of the norm of the gradient, which is used in the privacy calculation.

[11] Note that this differs slightly from the algorithm described in Abadi et al. [1], but is faithful to the corresponding implementation.

We can compute the sensitivity as the maximum change in the norm of the gradient due to the inclusion/exclusion of a single training example. If $\bar{\mathbf{g}}$ is the gradient averaged over one batch, and $\bar{\mathbf{g}}'$ is the average gradient when one training example in the batch (indexed by j, say) is changed, the maximum possible change in the norm of the gradient is

$$\begin{aligned} |\bar{\mathbf{g}} - \bar{\mathbf{g}}'| &\leq \left| \frac{1}{B} \sum_{i \neq j}^{B} [\mathbf{g}_i + \mathbf{g}_j] - \frac{1}{B} \sum_{i \neq j}^{B} [\mathbf{g}_i + \mathbf{g}'_j] \right| \\ &= \left| \frac{1}{B} [\mathbf{g}_j - \mathbf{g}'_j] \right| = \frac{1}{B} \sqrt{|\mathbf{g}_j|^2 - 2\mathbf{g}_j^T \mathbf{g}'_j + |\mathbf{g}'_j|^2} \\ &\leq \frac{1}{B} \sqrt{(CB)^2 + 2(CB)^2 + (CB)^2} = 2C \end{aligned} \tag{4.16}$$

This motivates the use of noise with standard deviation $C\sigma$ - C sets the effective scale of the gradient, and σ allows us to control the *relative* level of noise.

In practice, if we choose C too high, we are overestimating the true sensitivity of the algorithm, resulting in poor privacy guarantees (or poor performance for the same privacy). Therefore, choosing the correct clipping threshold C is an important hyperparameter optimisation step.

Slow learning is especially problematic because every iteration of SGD 'costs' some privacy budget. Using, for example, the privacy analysis from Abadi et al. [1], the effective privacy loss grows with $\sqrt{T}$ where T is the number of training iterations. This means that for the DP-SGD algorithm and their corresponding privacy budgeting procedure (the 'moments accountant'), σ would need to grow with $\sqrt{T}$ to provide the same ϵ, δ privacy guarantees for the overall procedure. Since GANs can have unpredictable learning behaviours without guaranteed convergence [146], it is especially challenging to train them using differential privacy.

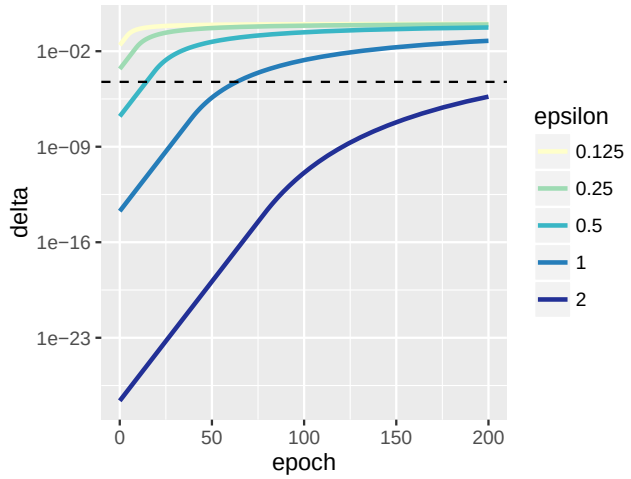


Figure 4.6: **Trade-off between ϵ and δ, and training epochs.** Probability (δ) of violating ϵ-differential privacy during training, for different values of ϵ using the moments accountant. Noise $\sim \mathcal{N}(0, 0.2^2)$ is used. The dotted line shows $\delta = 1/|D|$, where $|D|$ is the number of training examples.

Private generation of MNIST

As an initial proof of concept, we tested the DP-SGD algorithm on the RCGAN trained on MNIST digits, and evaluated using TSTR. We clipped gradients to 0.05 and added Gaussian noise with mean zero and standard deviation 0.05×2. For $\epsilon = 1$ and $\delta \leq 1.8 \times 10^{-3}$ (computed using the moments accountant [1]), we achieved an accuracy of 0.75 ± 0.03. Sacrificing more privacy, with $\epsilon = 2$ and $\delta \leq 2.5 \times 10^{-4}$, the accuracy is 0.77 ± 0.03. These results are far below the performance reported by the non-private GAN (Table 4.1), highlighting the compounded difficulty of generating a realistic dataset while maintaining privacy. For comparison, in Abadi et al. [1] they report an accuracy of 0.95 for training an MNIST classifier (on the full task) on a real dataset in a differentially private manner. This differs from our setting as we cast MNIST (unnaturally) as a multivariate time series classification problem, and moreover generate synthetic data for it.

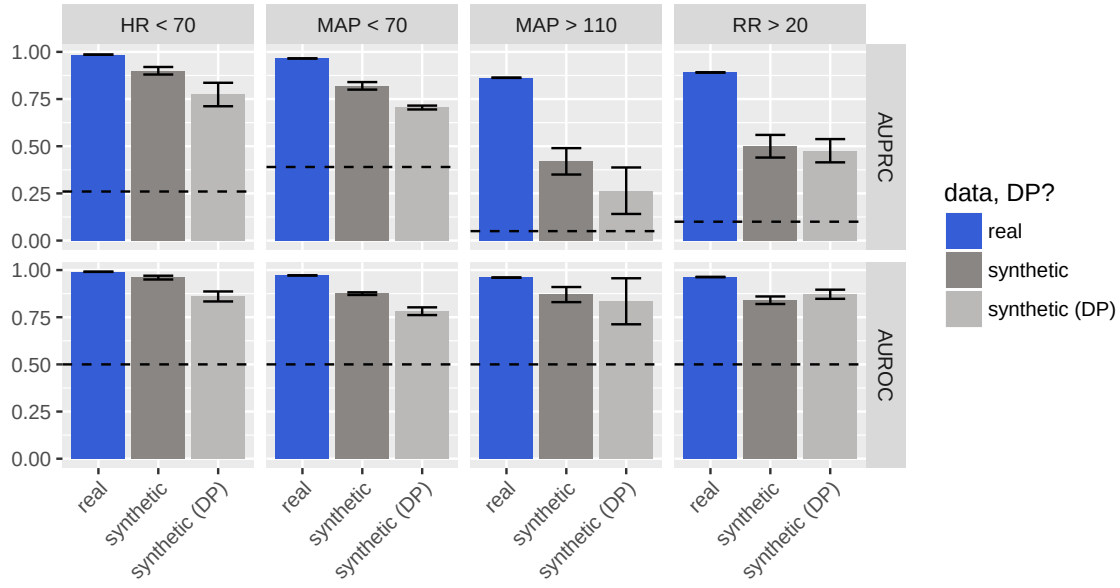


Figure 4.7: **TSTR results on four ICU prediction tasks**. We compare data generated by a non-private GAN (dark grey) and a GAN trained using differential privacy (light grey). The other three prediction tasks were used for model selection and are omitted here. We see that for some tasks, differentially private training of the GAN does not significantly impact the quality of the data measured by TSTR, but overall there is, as expected, a performance degradation. Here, $\epsilon = 0.5$ with $\delta \leq 0.9 \times 10^{-3}$. Clipping was set to $C = 0.1$ and the noise parameter $\sigma = 2$.

Private ICU data generation

We also apply the differentially private training procedure to the eICU setting, introduced in Section 4.4.3 The results are shown in Figure 4.7. For this case, we clipped gradients to 0.1 and added noise with standard deviation 0.1×2. In surprising contrast to our findings on MNIST, we observe that performance on the eICU tasks remains high with differentially private training, even for a stricter privacy setting ($\epsilon = 0.5$ and $\delta \leq 9.8 \times 10^{-3}$). Visual assessment of samples generated by the differentially-private GAN indicate that while it is prone to producing less-realistic sequences, the mistakes it introduces appear to be unimportant for the tasks we consider. In particular, the DP-GAN produces more extreme-valued sequences, but as the tasks are to predict extreme values,

it may be that the most salient part of the sequence is preserved. The possibility to introduce privacy-preserving noise which nonetheless allows for the training of downstream models suggests interesting directions of research in the intersection of privacy and GANs.

4.6 Conclusions and future work

The work described in this chapter highlights further questions and potential avenues for investigation, some of which are already underway in the research community.

- Why is it sufficient to sample latent points independently at each stage of generation? How would imposing (e.g. temporal) structure in the latent space impact the model, and could it be exploited in the context of conditional GANs?
- GANs are only one class of generative models. Other approaches, such as variational autoencoders [121] (potentially exploiting normalising flows [184]) have also been turned to the task of generating synthetic time series [57] and could be exploited in this domain as well.
- The methods described here generate data streams with a fixed (identical) sampling frequency. This is not generally the case in medical settings, so to generate truly realistic medical data, irregular sampling frequencies and missing data should be handled.
- Evaluation methods for GANs is a growing field. Some approaches were highlighted in Section 4.4, but more continue to be proposed. For example, Olsson et al. [171] borrow ideas from skill rating in games like chess to rank GANs.

CHAPTER 5

MODELLING INTENSIVE CARE

> Prediction is very difficult,
> especially about the future.
>
> *Various sources*

Individual contributions *This chapter describes work done as part of a collaboration between ETH Zurich and Inselspital Bern. From ETH, Xinrui Lyu, Matthias Hüser, and Cristbal Esteban worked alongside me, with our supervisor Gunnar Rätsch, and Martin Faltys and Tobias Merz from Inselspital. While all were engaged in discussions and design of the project, individual components were implemented by different people. Specifically, Martin Faltys was responsible for initial variable identification and provided clinical consultation throughout, alongside Tobias Merz. Xinrui Lyu was responsible for artefact removal, patient filtering, and pivoting and merging tables. Matthias Hüser was responsible for feature and label extraction, imputation, and training the model. I was responsible for processing pharma variables, implementing endpoint identification and variable merging, data extraction from MIMIC, and providing supervision and project guidance throughout. Collaborators from ETH in Basel, including Thomas Gumbsch, Michael Moor, and Bastian Rieck were also involved in discussions towards the end of the project, particularly about new evaluation techniques, including the event-based evaluation.*

5.1 The Intensive Care Unit

Intensive (or critical) care is a relatively modern speciality within medicine concerned with the treatment of critically ill patients. It was observed as early as the 1850s by Florence Nightingale [221] that especially sick patients would benefit from being situated closest to the nursing station, such that they could be monitored more closely. In the 1920s and 1930s, special units were developed [232] to provide additional monitoring and recovery care for post-operative patients. The idea of intensive monitoring is core to the philosophy of intensive care. Combined with the need to provide specialist care, with a focus on organ support, this leads to the importance of the *intensive care unit* (ICU), which can be found in larger hospitals. The ICU is staffed by a dedicated team of ICU nurses, specially trained intensive care physicians, and other support staff. Patients in the ICU are monitored 24 hours per day, 7 days a week, with a dedicated nurse at their bedside. Some ICUs allow for one nurse per two patients, provided those patients are not intubated [30]. Therefore, intensive care medicine is also resource-intensive, and constitutes a significant cost - a study from the Welsh NHS in 2011/2012 reported that ICU beds cost on average 4 times as much as ward beds [225]. However, the ICU provides a level of care unmatched by traditional wards with concomitant reductions in mortality [116], and the ratio of ICU beds to hospital beds is predicted to grow [221].

5.1.1 Machine learning and the ICU

Data-driven models are already used routinely in intensive care, the most famous of which being the APACHE system [123]. APACHE (and its variants)

uses physiological values from the patient's first day in the ICU, along with medical and surgical disease categories, admitting location and co-morbidities, to produce a mortality estimate. APACHE and related scores can be used for risk stratification of patients on admission, and have been used to benchmark ICU performance [83], although APACHE-II was shown to consistently overestimate mortality in a Scottish ICU cohort [84], indicating the need to calibrate models by geography (APACHE was developed on a US cohort). Moreover, as noted by Kelly et al. [116], the improving quality of intensive care over time has also necessitated recalibration.

The combination of intensive monitoring producing extensive data, the importance of time-sensitive decision-making in critically ill patients, and the interest in reducing costs makes the ICU an attractive candidate for the use of machine learning and data science. The existence of open-access ICU datasets such as MIMIC-III [110] (discussed already in Chapter 3) has also enabled machine learning researchers and other data scientists to develop models on ICU data without the need for clinical collaborators with access to data. Initiatives such as 'critical care datathons' [2] have also raised the profile of the intensive care as a venue for data science.

Perhaps unsurprisingly, much attention from the machine learning community in ICU has focused on mortality prediction [9, 16, 70, 79, 142, 149, 223, 244], although other questions such as diagnosis prediction and classification [38, 144, 176, 177, 182] and length of stay prediction [85, 215, 238] are also common themes. Various disease or condition-specific applications have also been developed, for example predicting sepsis or septic shock [31, 71, 202, 212], heart failure or cardiac arrest [37, 117], post-transplant complications [55], and respi-

ratory failure [99]. In this chapter we also focus on a specific condition, namely circulatory failure.

5.2 Circulatory failure prediction

Our core prediction task is to identify whether a patient's circulatory system will deteriorate in the near future. This deterioration manifests as circulatory shock, a condition in which insufficient oxygen is delivered to (and utilised by) the patient's tissues. Circulatory shock is a common occurrence in ICU patients, with many aetiologies and categories. These types of shock can be differentiated by the source of the circulatory insufficiency - cardiogenic shock points to the heart itself, whereas in distributive shock, cardiac function and output may be normal, but due to peripheral issues oxygen delivery nonetheless fails. The most common form of distributive shock is septic shock, occurring in 62% of shock cases in one study [222]. Septic shock, as part of sepsis or systemic inflammatory response syndrome, contributes to poor outcomes. Mortality rates in sepsis depend on the specific definition of sepsis (a topic of ongoing debate), but consensus points towards sepsis *with* shock having mortality rates in excess of 40% [196]. The challenge of shock within and outside of sepsis is that tissue hypoperfusion resulting from unresolved circulatory failure leads to further organ failure, and potentially multiple organ dysfunction syndrome, one of the leading causes of mortality in the ICU. Even more alarmingly, the treatment for severe circulatory failure can itself lead to organ failure, as other organ systems struggle to compensate for intensive circulatory support [76].

Early intervention to prevent widespread damage as a result of circulatory

failure is therefore critical. Treatment of circulatory shock involves restoring blood pressure and providing support to crucial organ systems while the underlying cause is ascertained. Fluids can be administered, especially in the case of hypovolemic shock, to support pressure. Vasopressors - drugs which cause constriction of blood vessels, can provide fast-acting blood pressure support with a low half-life. Continuous infusion of vasopressors such as norepinephrine at controlled dosage allows nurses and other ICU caregivers to control the patient's mean arterial pressure (MAP).

However, identifying which patients are at risk of deterioration is challenging. Existing track-and-trigger systems for identifying deterioration in ward patients have been criticised for lacking reliability and utility [68]. A 2016 study [162] using data from over 9000 ICU patients reported an AUROC of $0.61 - 0.88$ in predicting a set of deterioration endpoints such as sepsis, unplanned intubation, and significant haemorrhage in the next 24 hours. Work predicting shock *outside* of sepsis is rare, making it challenging to identify an existing state of the art, but related tasks have been studied. Within sepsis prediction - not necessarily with shock - performance ranges from AUROC of 0.72 within 6 hours [202], 0.83 within 3 hours [31] to 0.909 for any post-operative sepsis [212], in varying cohorts. Predicting the onset of vasopressors can implicitly capture the onset of circulatory shock. [207] report an AUROC of 0.77 using either LSTMs or CNNs predicting the onset of vasopressors within 10 hours, and [235] achieve an AUROC of 0.92 on imminent (within 2 hours) vasopressor need. The 2009 annual PhysioNet challenge focused on predicting acute hypotensive episodes in ICU patients, defined as regions where mean arterial pressure is below 60 mmHg. The winning entry achieved an accuracy of 92.5% with a recall of 92.8% predicting these hypotensive episodes up to an hour in

Table 5.1: **Definition of circulatory failure.** The level of severity is set by the strength of the vasoactive support given through drugs.

(a) The patient's circulatory state depends on the presence of shock (MAP $\leq$ 65 mmHg, or vasoactive drugs) and hyperlactataemia (lactate $\geq$ 2 mmol/L). Ambiguous states exist when only some criteria are met, or there is missing data.

patient state	blood pressure	lactate
stable	MAP $>$ 65 mmHg *and* no vasoactive drugs present	$<$ any value
circulatory failure	MAP $\leq$ 65 mmHg *or* (non-exclusive) vasoactive drugs present	$\geq$ 2 mmol/L
ambiguous	MAP $\leq$ 65 mmHg *or* (non-exclusive) vasoactive drugs present	$<$ 2 mmol/L *or* missing

(b) Drugs used for cardiovascular support in the presence of shock, graded by level.

level	drugs
1	any dose of dobutamine, milrinone, theophylline, levosimendan
2	$\leq 0.1\mu$g/kg/min of epinephrine or norepinephrine
3	$> 0.1\mu$g/kg/min of epinephrine or norepinephrine, or any dose of vasopressin

advance, however based on the cohort, the positive prevalence was 41%. This is quite different to our setting, where positive events occur approximately 3% of the time.

5.2.1 Definition of circulatory system failure

We are interested in predicting circulatory system deterioration. Deterioration specifically requires the patient to move into a state of worse circulatory system function. This requires a definition of circulatory system function, which we define according to our clinical collaborators in three levels of increasing severity in Table 5.1.

The blood pressure criterion notably includes a check for the presence of vasoactive drugs. This is necessary because caregivers will intervene on a patient

with deteriorating MAP with therapy (vasoactive drugs) which will make their MAP appear deceptively 'normal' [178]. Thus, the endpoint we seek to predict is a mixture of underlying patient state and the actions of clinicians, which are invariably interlinked. Thus, we consider the presence of these vasoactive drugs to be sufficient to trigger the blood pressure criterion, regardless of the actual MAP. However, elevated lactate (hyperlactataemia) indicates that the circulatory insufficiency is such that tissue hypoperfusion has become significant, and is therefore a necessary condition.

The vasoactive drugs we consider are dobutamine, milrinone, theophylline, levosimendan, epinephrine, norepinephrine, and vasopressin. We can further identify *levels* of circulatory failure based on which of these drugs are present, with increasing severity. The levels with corresponding drug conditions are shown in Table 5.1b. In the event that a patient is receiving multiple vasoactive drugs (and meets the other criteria), we can define the level of circulatory failure by the *highest* level they satisfy. There are two ambiguous states arising from the definition above:

1. Hypotension in the absence of hyperlactataemia is indicative of *early-stage* circulatory dysfunction, and is neither a stable (level 0) patient, nor one with circulatory failure (levels 1 to 3).
2. Lactate measurements require an arterial (or venous) blood gas analysis to be performed, requiring a blood draw, and are therefore not continuously recorded. We have to do some form of missing-data imputation (described in Section 5.3.4) to make statements about the lactate level at most time-points. In the cases where imputation is not performed (for example, no historical lactate values for a patient, or the time since the last

test is beyond a threshold), the state is once again ambiguous as we cannot distinguish real circulatory failure from early-stage failure.

We can then use the above definition to label every time-point of a patient's stay into one of the four levels, or as an ambiguous state. To reduce fluctuations in MAP or lactate measurements resulting in spurious false positive or false negatives, we additionally require that the MAP and lactate conditions be true for **30 minutes in a 45-minute window**. Thus, neither a single measurement of MAP = 64 mmHg in an otherwise stable patient, or a single measurement of MAP = 66 mmHg in a patient with clear circulatory failure will result in an incorrect label.

5.2.2 Definition of prediction task

Given a labelling of a patient's time-series into the above states, we can formulate the deterioration problem simply by asking if a patient will move to a worse state (than they are currently) in the near future. More specifically, if a patient is in state s_0 at time t_0, a deterioration in the next T hours means the patient is in state $s > s_0$ at *some* point in the interval [$t_0 + 5$ minutes, $t_0 + T$ hours]. We require a five minute lead time to give clinicians enough time to potentially intervene. We choose $T = 8$ as this is the length of a shift, and increases the baseline prevalence of deterioration events.

Initial experiments indicated that predicting further deteriorations (for example, from level 1 to 2) is markedly more challenging than predicting *any* kind of deterioration from stability. Leaving the question of fine-grained and subsequent predictions to future work, we therefore focus on predicting the transi-

tion from no circulatory dysfunction, which we refer to as *stability*, to any level of circulatory failure. That is, we consider only where $s_0 = 0$. If a patient is in another state (for example, level 2, or an ambiguous state) we do not label that timepoint.

5.3 Data preparation and processing

Preparing raw data from an ICU patient data management system (PDMS) for use in a machine learning system is a substantial task. In this project, we devised a processing pipeline comprising several modularised steps, to enable distribution of the tasks among several researchers. Those responsibilities are described at the start of this chapter.

5.3.1 Data and patient filtering

We apply several kinds of quality filtering and patient subselection to the data. The procedure is shown in Figure 5.2. We removed patients with ECMO/Impella use because this introduces different physiology and should be exempt from our early warning system. Ideally we would remove just the periods during which they were on ECMO/Impella, but use of ECMO/Impella was recorded in nursing notes which could not be matched to specific timepoints, so the patients were entirely excluded. This applied to 290 patients only.

We applied various kinds of artefact removal. These focus on correcting human error in timestamp entry, fixing erroneously labelled variables (specifically, arterial blood gas samples which are labelled as venous and vice-versa),

and applying cutoffs according to physiologically plausible values. These upper and lower bounds were defined for each variable by a clinical collaborator. Figure 5.1 shows an idealised case of a pulse oximeter reporting an erroneous value of SpO2 due to noise, potentially caused by the patient moving around.

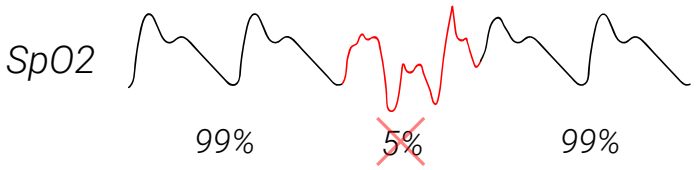


Figure 5.1: **An illustration of noise in the signal from a pulse oximeter.** The noise results in an erroneous value of SpO2. Physiologically implausible values such as 5% for SpO2 are removed during artefact removal.

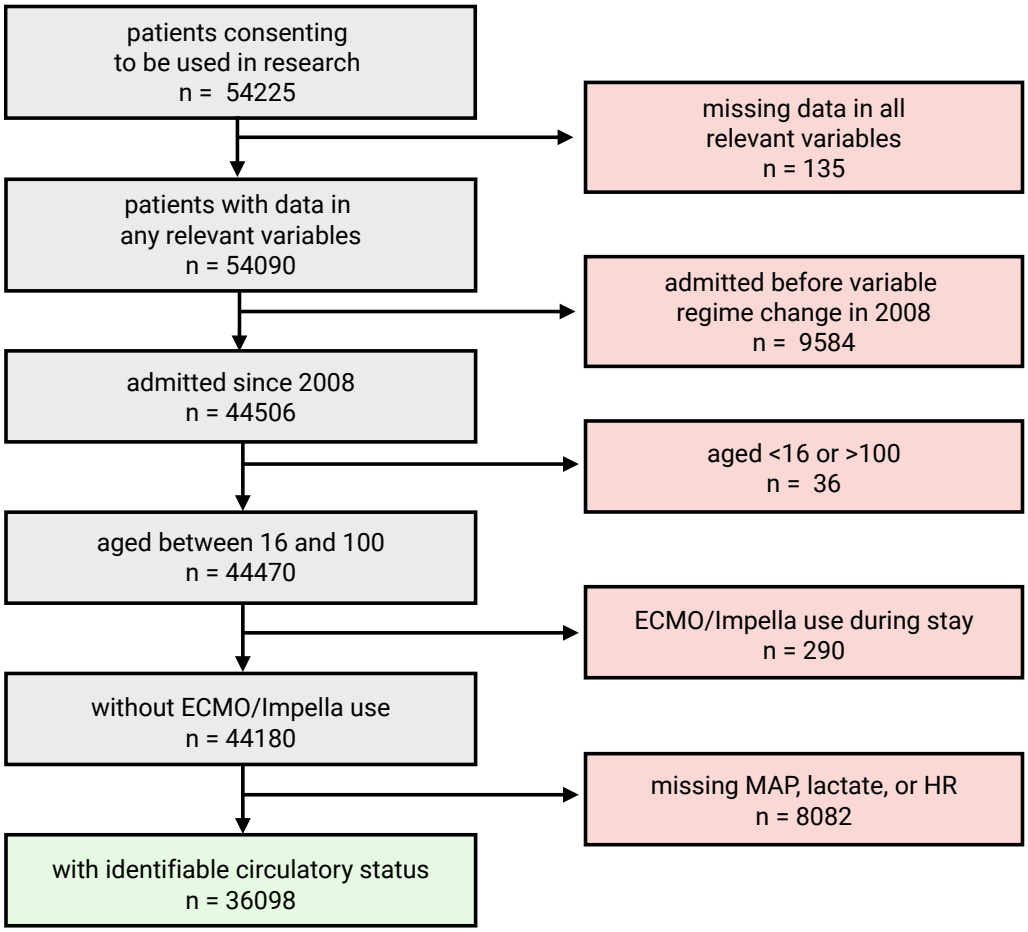


Figure 5.2: **Flowchart showing patient inclusion and exclusion steps.** In the end, we use a set of 36098 patients to build and evaluate the predictive model. Of thee patients, 9801 have instances of circulatory dysfunction.

Table 5.2 shows statistics about the patient cohort.

5.3.2 Converting drugs to flow rates

Patients can receive drugs in multiple ways: either by continuous infusion, or in a bolus (e.g. injection or tablet). To standardise the semantics of pharmaceutical variables, we attempted to convert all drug information into a "flow rate". In the database, drug-related variables have status codes which indicate if the measurement corresponds to the beginning/end of an infusion, a change in rate, or an instantaneous bolus. (These status codes are occasionally incorrect). The value corresponding to the measurement is typically the dose given since the last record. The units used to record doses for a drug occasionally change in this database, so it is necessary to also cross-reference a table of unit changes in case the value needs to be scaled to ensure consistent units.

A patient may receive the same drug through multiple simultaneous infusions, in which case the value is the dose given *in that infusion* since the last record *in that infusion*. Identifying the total flow rate of a given drug thus requires merging flow rates across infusion channels.

To convert boluses to flow rates, one of our clinical collaborators provided effective acting periods for each drug. In a simplification of pharmacodynamics, we assume that the drug acts as a constant flow over the acting period, where the rate is given by the total dose of the bolus divided by the acting period. For fast-acting drugs such as norepinephrine, this acting period is 5 minutes, which is the most granular time-delta considered here, so these drugs are effectively 'instantaneous'.

Table 5.2: **Patient demographics.** For length of stay, we do not consider data from patients after their 28th day in the ICU (this applies to 137 patients only). We define length of stay as the time between the first and last recorded heart rate measurement. Note that patients can have more than one APACHE diagnostic group. The 'other' category includes trauma, gastrointestinal, sepsis, metabolic, haematological, orthopaedic, gynaecological, and renal (all below 5%).

Sex	
Male	62.38%
Female	37.62%
Age (years)	
Median (Mean)	65 (62)
Range	16-98
Length of stay (days)	
Median (Mean)	0.93 (2.09)
Range	0-84
Admission type	
Emergency	59%
Not emergency	39%
Surgical	
Yes	49.85%
No	47.13%
APACHE diagnostic group	
Cardiovascular	
Surgical	13.83%
Nonsurgical	7.27%
Neurological	
Surgical	7.46%
Nonsurgical	13.30%
Respiratory	
Surgical	1.47%
Nonsurgical	4.12%
Trauma	
Surgical	0.87%
Nonsurgical	3.66%
Other	15.28%
Unknown	42.39%
Circulatory dysfunction	
Patients with events	25.42%
Mean events per patient	4.12
Mean event duration	5.3 hours
Mean time to first event	20.5 hours
Mortality	4.96%

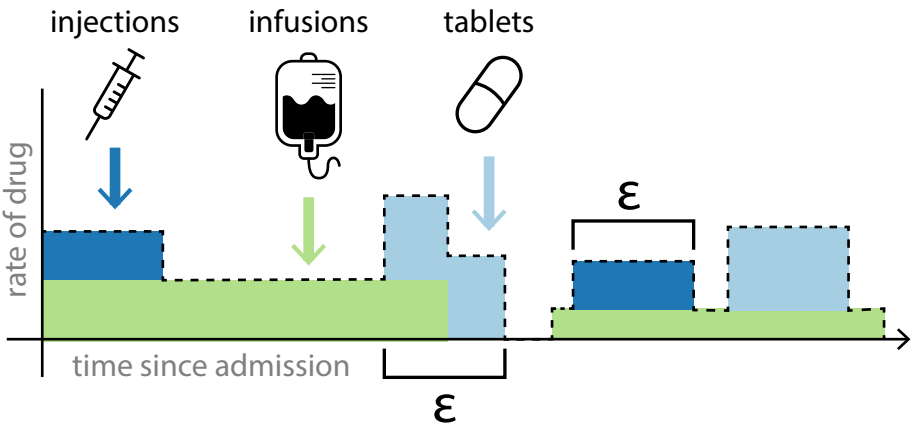


Figure 5.3: **A schematic showing how the same drug can be administered through multiple delivery routes simultaneously.** Most commonly, a patient receives multiple infusions at the same time. For delivery routes such as injections and tablets, we convert the dose into an effective 'rate' by considering it to be delivered continuously over a time period ϵ, which is defined for each drug based on clinical prior knowledge of its acting period.

Table 5.3: **Examples of how the same variable is recorded in multiple ways.** To reduce redundancy in our features and imputation needs, we merge these 'duplicated' variables. Note that ZVD is Zentraler Venendruck - central venous pressure.

PDMS name	variable name
Noradrenalin 1mg/ml	noradrenaline
Noradrenalin 100 µg/ml Perfusor	noradrenaline
Noradrenalin 10 µg/ml Bolus	noradrenaline
ZVDm	central venous pressure
M_ZVD	central venous pressure
Temp Kern	temperature
Temp Axilär	temperature

5.3.3 Variable merging

The PDMS at Inselspital contains many redundant variables. Since this database is used operationally and was not designed for secondary use of this kind, many semantically identical variables are duplicated, presumably for the purpose of convenience. In Table 5.3 we show examples of entries in the PDMS corresponding to the same variable.

We would like to merge such variables for two reasons:

1. To develop a model which is not overly specific to the ICU in Bern
2. To reduce the need to do missing-data imputation

In some cases, variables are not *completely* identical, but we consider them sufficiently similar to merge, for example various ways of measuring body temperature (shown in Table 5.3). In the event that there are multiple simultaneous measurements of a real-valued variable, we take the median value.

Drugs are handled separately: since drugs are treatments given to a patient and not measurements of some underlying state, we have to sum over identical drugs to produce an overall dosage. To help further reduce the dimensionality of the input data, our collaborator specified which drugs are *functionally* similar. For example, we merge the drugs atenolol, bisoprolol, metoprolol, and others into an aggregate variable which indicates if any beta-blockers are present. We do similarly for ACE inhibitors, loop diuretics, antibiotics, opiates, and so on. For some classes of drugs, like opiates, we are able to define effective equivalent doses and produce an overall effective dose. For anti-epileptics, we simply take the *number* of drugs present. In other cases, we use binary indicators (as for beta-blockers).

By doing this, we can merge 82 antibiotics, each appearing in 1.1% of patients (on average), to a single indicator, now present in 62.5% of patients. Overall, we go from 708 variables appearing each in (on average) 9.8% patients, to 198 each appearing in 32.4%. Put another way, with 708 variables, each patient has 31.6 measurements of each (on average). With 198, this becomes 138.2.

5.3.4 Imputation

In this section we distinguish between two types of imputation: imputation for generating labels, and imputation for generating (eventually) features. To generate labels, since these are not required to deploy the model, we can use any information, including information from 'the future', for imputation. To impute for feature construction, it is important that we neither use information from the future, nor information from the held-out test set(s).

Imputing lactate for endpoint identification

As mentioned in the endpoint definition (Section 5.2.1), lactate is critical for the identification of circulatory failure. However, lactate is not continuously measured and measurements are recorded on average every 5.5 hours as part of a blood gas analysis, which may be ordered either routinely or as deemed necessarily by a clinician. This means that the measurement of lactate depends on a human's subjective assessment of how variable the lactate may be. For this reason, we (in discussion with clinical collaborators) devised an imputation strategy for lactate depending on whether or not the patient's lactate is 'normal'. We define abnormally high lactate to be lactate above 2 mmol/L, the same cutoff used in the endpoint definition (Table 5.1). For notation, we call this cutoff l_C.

For a pair of lactate values, l_1 and l_2 recorded at timepoints t_1 and t_2 (for a specific patient), we use different imputation strategies depending on the perceived state of the patient.

If $l_1 < l_C$ and $l_2 < l_C$, *or* $l_1 \geq l_C$ and $l_2 \geq l_C$, we consider the patient 'stable', and perform linear interpolation without time restriction. The assumption here

is that if the lactate is consistently above or below the threshold, the clinician may perceive the patient to be stable, and order tests less frequently. We follow the assumption that in the intervening time, the lactate will remain either above or below the threshold as it was before, which can be achieved by linear interpolation. Once again, we err on the side of modelling lactate dynamics very simply, even if lactate clearance may truly follow more complicated dynamics.

If the patient is unstable, in that l_1 and l_2 are not the same side of l_C, then we are more conservative. In this case, since we (as above) don't know the exact lactate dynamics, we instead forward-fill l_1 for up to 3 hours, and backwards-fill l_2 for 3 hours. This reflects our belief that if a patient is in a normal or abnormal state at time t we are only confident for $\pm$ 3 hours that their state has not changed. If the time between t_1 and t_2 is less than 6 hours (so that these forwards/backwards-filling windows would overlap), we instead linearly interpolate during the 6-hour window.

If l_0 is the first lactate measurement, and l_ω is the final, we handle the edge cases as follows:

- If $l_0 < l_C$, we backwards-fill.
- If $l_0 \geq l_C$, we backwards-fill for up to 3 hours.
- If $l_\omega < l_C$, we forwards-fill.
- If $l_\omega \geq l_C$, we forwards-fill for up to 3 hours.

This uses the same logic as before.

If a patient has no lactate measurements at all, we do nothing, and they are eventually filtered out for being in a permanently ambiguous state.

General imputation

As part of data processing for feature extraction, we project all measurements onto a regularly-sampled time grid with 5-minute interval. This introduces many missing values, which must then be imputed.

For feature (pre-)construction, we perform a mixture of forward-filling and linear interpolation in a variable-wise fashion, excluding drugs as they do not require imputation. As in the case of lactate, we assume that the measurement frequency is informative for the time-scale over which a variable is expected to change.

To reflect this, we first calculate the median interval (m_v) between subsequent measurements for *each* variable v (using the available training set), as well as the interquartile range of the sampling interval (r_v). From this we define an interval $\Delta_v = m_v + 2r_v$.

We then impute variable v according to this scheme:

1. Assuming the last observed value of v at t is x_t, forward-fill with x_t up to $t + \Delta_v$.
2. Between $t + \Delta_v$ and $t + 2\Delta_v$, linearly interpolate between x_t and $\tilde{x}$. $\tilde{x}$ is defined as the median value of v over $[t - 2\Delta_v, t]$, excluding values introduced by imputation.
3. After that, forward-fill indefinitely.
4. In all cases above, the imputation strategy is restarted after every real observation of v.

The above strategy is depicted in Figure 5.4.

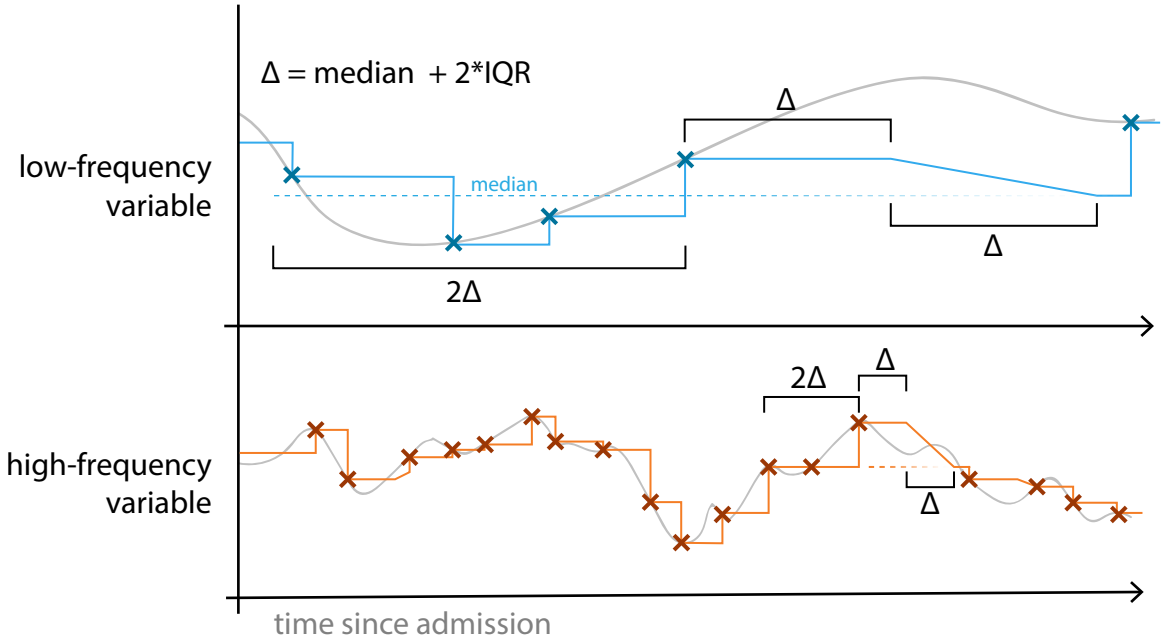


Figure 5.4: **A schematic depicting the imputation strategy for time-varying variables.** For each variable v, a timescale Δ_v is computed as the median of the variable's sampling interval plus twice its interquartile range. Imputation then forward-fills for Δ_v, before decaying over a time Δ_v to the median value over the time-period $2\Delta_v$ before the last observed measurement. After that, indefinite forward filling is applied.

For static variables such as age and APACHE group, values are imputed as the median or mode over patients in the training data.

5.4 Modelling

Since our data comes in the form of a dense matrix of measurements, we need to produce a representation amenable to the classification task at hand. A first choice may be to simply leave the data as-is, and treat it as a time-series modelling problem, for example using a recurrent neural network (as discussed in chapters 2 and 3). Recurrent neural networks learn to transform input data

such that their internal state holds a representation of the data observed so far. This representation should be optimised for the classification task, effectively representing different types of patient trajectories (deteriorating or remaining stable) in an easily separable way. However, naïve application of recurrent neural networks for this task has a significant downside, one which cannot even be overcome by the methods described in chapter 2. The problem is: since we discretise our time-series into five-minute intervals, a prohibitively large number of time-steps need to be considered in order to observe even *two* real measurements from infrequently measured variables. For example, the median sampling interval for neutrophil count is 1 day, so observing two subsequent measurements would require 577 timesteps to be considered. Even using long-memory RNNs, sequences of this length are challenging, and computationally prohibitive to train.[1]

In this section we describe the extracted features and the classification model.

5.4.1 Feature extraction

In total, we extract 3684 features from four classes. However, we found that using 1000 would be sufficient and only marginal improvements could be attained using the full set of features. We consider these 1000 in the reference model. Listed below are the feature classes, with their contributions to the full and reference set of features in brackets:

[1] In a preliminary analysis using a LSTM with engineered features, it achieved an AUROC of 0.923 and AUPRC of 0.430, which is comparable (but inferior) to the performance of even the limited-variable lightGBM model we consider further, which achieves AUROC 0.929 and AUPRC 0.445 - examined in later sections.

1. **Multiscale**: [3354,855] features calculated over the recent past of the patient's time series
2. **Measurement**: [226,116] features capturing aspects of measurement otherwise lost during imputation
3. **Endpoint-derived**: [98,25] features related to the definition of circulatory dysfunction
4. **Static**: [6,4] time-invariance demographic and admission information

Each class is described in more detail below.

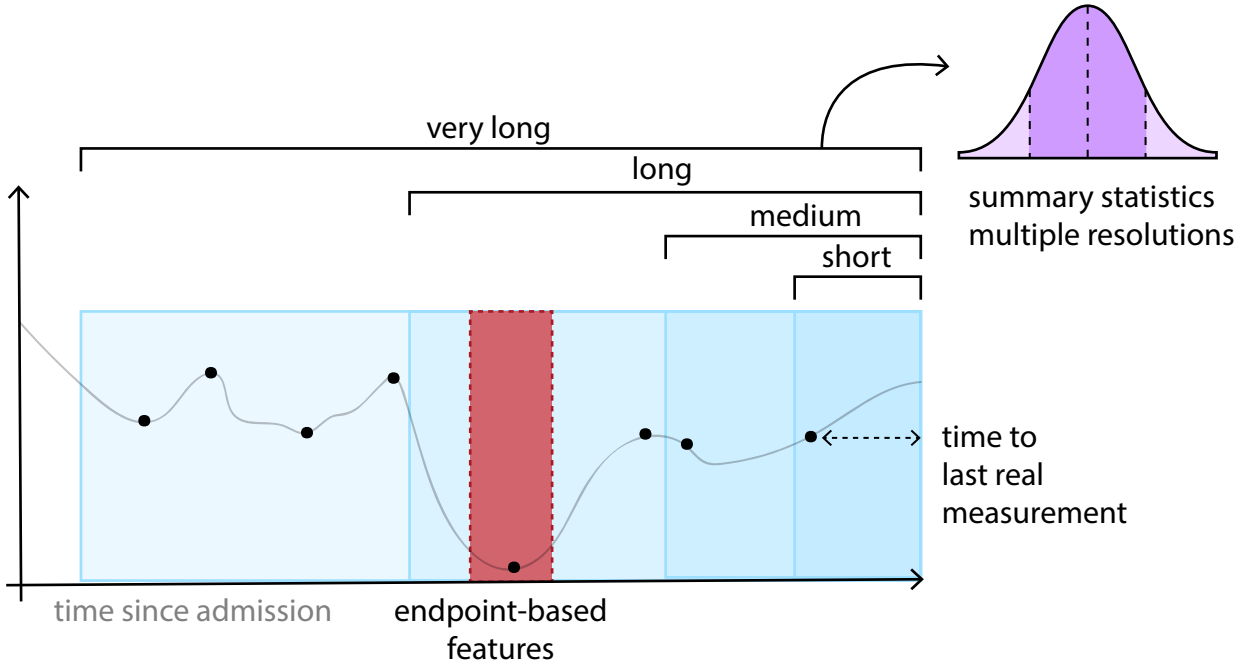


Figure 5.5: **Feature classes extracted from time-series data.** Using overlapping windows of increasing size, we summarise the time series at multiple resolutions using 5-point summary statistics (minimum, maximum, median, interquartile range, trend). We record the time since the last non-imputed measurement, as well as statistics about the fraction of the patient's time spent in endpoints.

Multiscale features

The simplest way to represent time-series data would be to take T timesteps of (potentially partially imputed) observations of d variables and concatenate them into a vector of length Td. This presents two immediate problems:

1. T can grow very large if we use small timesteps. With a sampling interval of 5 minutes, even capturing even the last day of information for a patient would require 288 timesteps. Considering d is 198 (after merging variables, see Section 5.3.3), the feature representation may become computationally unwieldy. Moreover, for variables recorded at a lower sampling frequency, the majority of values in the vector will be the same, resulting in an overparametrised model.
2. If the model class assumes each dimension in the input feature is independent (such as a linear model), this representation needlessly throws away useful information about how the data is structured temporally.

In principle, a more expressive model class (such as a deep neural network) is not affected by the second issue. Such a model could, with sufficient data (and successful optimisation) learn to combine inputs in such a way as to extract temporal features such as trends, assuming these features are relevant for classification.

However, we decided to rather first design features based on clinical prior belief. What we do is to define a set of M *horizons* for each variable, and summarise the values observed in the time window defined by the horizon. Extracting S statistics from each window, this reduces the dimensionality of the

feature vector from (naively) Td to SMd. By defining the horizons appropriately, we can implicitly capture T hours of data without paying a cost in the number of parameters. In practice, we use $M = 4$ and $S = 5$.

Horizons Since variables are recorded (and vary) at different rates, it is important to use window sizes which capture the range of variation in this sense. For blood pressure, changes in the last 30 minutes could be important, whereas lactate (which is measured on average every 5.5 hours), there is unlikely to be any change. Rather than producing redundant or excessive features, we define different sets of resolutions for different variables. We first categorise each variable as low, medium, or high frequency based on its median sampling interval[2], and then define four window sizes (τ_i) for each category, see Table 5.4.

Table 5.4: **Categorisation of variables by frequency and corresponding window sizes.** Each variable is categorised as high, medium, or low frequency based on its median sampling interval. Four windows are defined for each category in order to define multiscale features.

category	sampling interval	τ_1	τ_2	τ_3	τ_4
high	$\leq$ 15 minutes	30 minutes	1 hour	4 hours	12 hours
medium	(15 minutes, 8 hours]	12 hours	24 hours	36 hours	48 hours
low	$\geq$ 8 hours	16 hours	32 hours	48 hours	72 hours

The windows are defined as the region $[t-\tau_i, t]$ at timepoint t, thus they overlap. This is shown by the overlapping blue boxes in Figure 5.5. By the horizons we have defined, this class of features implicitly captures up to 72 hours worth of data (from the slowest-varying variables), and allows for a summary of the patient's state with differing levels of granularity. To fully capture the data in these windows using concatenation would require between 144 and 864 (for 72 hours) features per variable. If the current timepoint t is less than 72 hours after

[2]We reuse the statistics already computed during imputation for this purpose.

the patient's arrival to the ICU, the window is simply clipped - so in the edge case of $t = 5$ minutes, all windows are identical.

We additionally include the median value of the variable over the patient's entire stay (until the current time point).

Statistics In each window, we compute a set of S statistics. In this work we choose to use a five-point summary (hence $S = 5$): over a window of length τ for a given variable v, we extract the median, the minimum, the maximum value, a 'trend' proxy, and the interquartile range (IQR) over the interval. In principle the trend would be the slope of a linear fit to the data in the window, but we approximate this (for computational reasons) by simply taking the difference between the first and last values in the window. We use median and IQR rather than mean and standard deviation since, despite best efforts to remove them, artefacts producing outlier values will likely exist in the data. Minimum and maximum are of course susceptible to such errors, but taking these statistics of data is common practice in ICU scoring (see e.g APACHE-III [123] and SOFA [220]). We also believe that trend information will be informative for identifying circulatory failure, although fitting a trend to nonlinear data can be misleading.

This is performed in each window τ_i for each variable, excluding categorical or binary-valued variables. For these, we extract simply the mean or mode.

Measurement features

Since missing data imputation discards information about when and how often certain measurements were taken, and this information can be informative of patient state [4], we introduce a feature class to attempt to recapture it.

For each variable, we record:

1. The measurement 'density' so far: the total number of (real) measurements divided by the time since admission.
2. The time since the last real measurement (shown in Figure 5.5).
3. The current imputed value.

Another feature we include in this class is the time elapsed since the patient was admitted to the ICU, which turns out to be quite important.

Endpoint-derived features

Clinical experience indicates that prior history of circulatory failure is indicative of increased risk of future circulatory failure. For this reason, we build a special class of features which take into account a patient's historical (during their stay) circulatory status, as well as derived features based on the endpoint definition. Although we have already labelled the data with presence of endpoints of the various types, we can't use this information during feature construction as it potentially relies on unavailable information in the form of interpolated lactate measurements. We therefore build endpoint-*related* features built from subcomponents of the endpoint, all taken as binary indicators. For example,

since presence of vasoactive drugs (listed in Table 5.1b) is a component of the endpoint definition, we construct features based on it in three classes:

1. **Current**: for example, whether or not vasoactive drugs are present at the current time
2. **Time-from-event**: the time since the derived feature was last present, e.g. the time since patient was receiving vasoactive drugs
3. **Activity density**: over five horizons ($[12, 24, 36, 48, \omega]$ hours, where ω covers the patient's entire stay), the fraction of time the condition was true (e.g. the patient was receiving vasoactive drugs)

These feature classes are calculated for each component of the endpoint definition, such as whether MAP ≤ 65 mmHg, lactate ≥ 2 mmol/L, and the presence of the various vasoactive drugs.

Static features

Several important characteristics of the patient are static or vary slowly:

- Age (ICU stays do not typically last more than a few weeks).
- Sex.
- Height (we consider weight a time-varying feature, as it is often tracked during a patient's stay).
- Whether or not the patient was an emergency admission.
- Whether or not the patient was admitted from surgery.

- APACHE-III [123] patient group - this is not the APACHE *score* (which uses the first 24 hours of the stay), but the diagnostic category the patient is admitted under.

For the purpose of feature construction, these are simply concatenated to the rest of the features before use in the model. As APACHE group is categorical, it's represented as a one-hot vector of length 17.

5.4.2 Model

Given these features, we can use any classifier which takes a fixed input.

We elected in this work to use gradient-boosted decision trees implemented in the lightGBM library [114]. Ensembles of decision trees such as this are popular choices in competitive machine learning arenas such as Kaggle for their performance 'out of the box', being more robust to hyperparameter choices than deep learning approaches. Tree-based models also provide measures of feature importance, which can aid in interpretability, although here we elect to use Shapley values to identify important variables (Section 5.5.2).

The basic premise of gradient boosting [67] is to iteratively build predictors (trees) such that at iteration t, the new predictor is trained to minimise the *residuals* of the ensemble so far. This way, model complexity is controlled by limiting the number of boosting iterations.

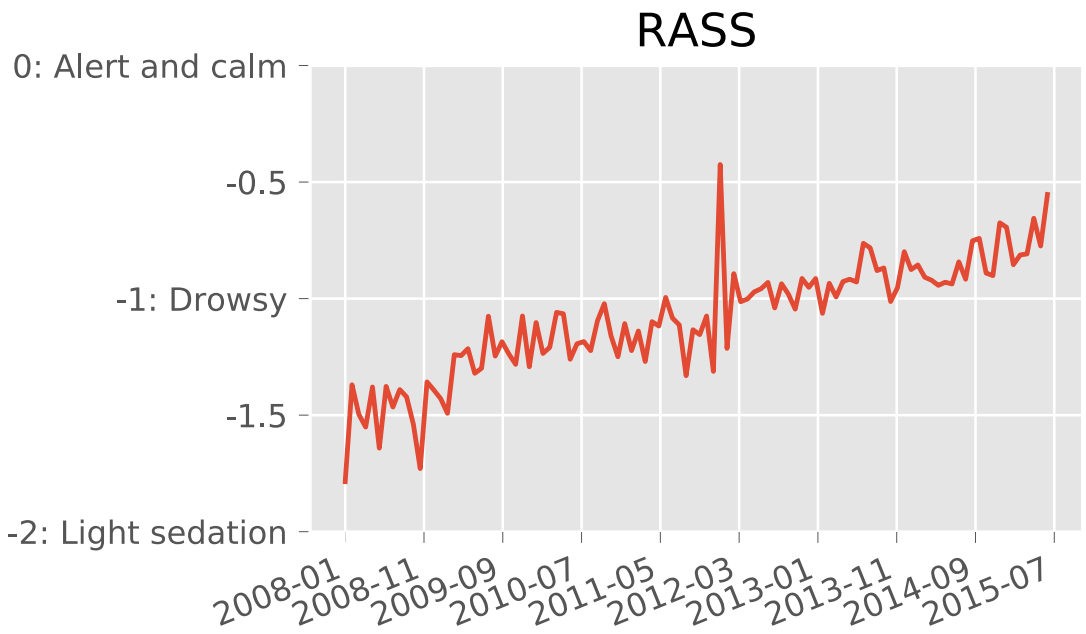


Figure 5.6: **Change of RASS over time**. The movement away from heavy sedation in critical care is reflected in the Inselspital data. Values show the mean RASS (Richmond Agitation-Sedation Score) in each month. This highlights that, even within an 8-year period, the practice of critical care is not stationary.

5.5 Experiments and results

5.5.1 Experimental setup

Developing a model on a retrospective dataset to be deployed in the present incurs unavoidable covariate shift between the training and test distributions. Here the shift is due to the change in medical practice and patient cohorts over time. For example, even within our dataset, we can observe the trend away from sedating patients, by looking at the average Richmond agitation-sedation score (RASS) over time (Figure 5.6). This observation reflects a known trend within critical care practice away from heavy sedation [10].

While traditional machine learning approaches assume the training and test set examples are independently and identically distributed, to do so in this case underestimates the generalisation error of the model (this is demonstrated in Table 5.6) when applied in a real setting. For this reason, we intentionally design the splits such that the training data contains the earliest patients (ordered by admission time), with the most recent 20% devoted to the validation and test splits. We additionally produce multiple 'replication' splits over the data, each using the same train/test split logic. Each fold covers a period of time of the same length, and allows us to ask questions about the behaviour of the generalisation error over time, explored in Section 5.5.5.

The setup is show in Figure 5.7.

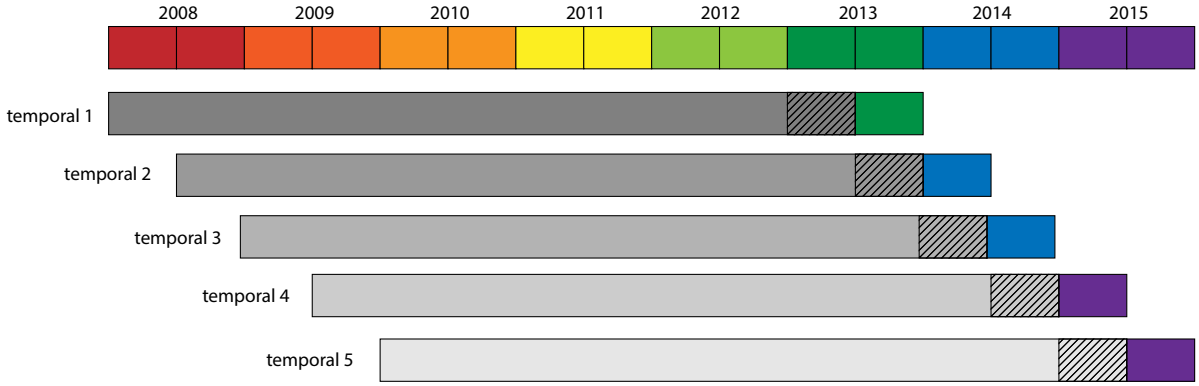


Figure 5.7: **Schematic showing how temporal splits are constructed.** The hatched region shows the validation set for a given split, and the coloured region is the test set. The same methodology is applied to each split - the validation and test sets comprise the patients in the final 20% of the split. Imputation parameters are computed on the training set, and the validation set is used for model selection.

5.5.2 Choosing models

Feature importance

Before we run analyses on our model, we must specify which model we mean. We aim to define a larger `reference` model, and a smaller model using just the most important variables.

In order to identify which features (and by extension variables) are important for the model, we use the SHAP (SHapley Additive exPlanation) approach [140], specifically exploiting the TreeSHAP approach for estimating Shapley values from tree-based models [141], which is implemented in lightGBM. Briefly, Shapley values are a solution to the problem of distributing winnings to participants in an n-player game based on their contributions [193], allowing for the possibility that some participants have redundant contributions. This idea has been applied to the problem of identifying which features in a model are responsible for contributing to a particular prediction [140]. To produce overall feature importances (since SHAP values are per example), we take the mean *absolute* SHAP value for each feature. To then define the important *variables*, we take the lowest rank of all the features derived from that variable.

As mentioned in Section 5.4.1, we have 3684 features. Early investigation indicated that many of these are (unsurprisingly) uninformative, assessed with SHAP values. Therefore, to proceed with a slightly simpler model, we restricted to a set of 1000 features. This is the `reference` model we use in the rest of the analyses. Of these 1000 features, 855 are history features, 116 are measurement-related, 25 are endpoint-related, and 4 are static. Figure 5.8 shows the distribution of (log) feature importance for the four classes.

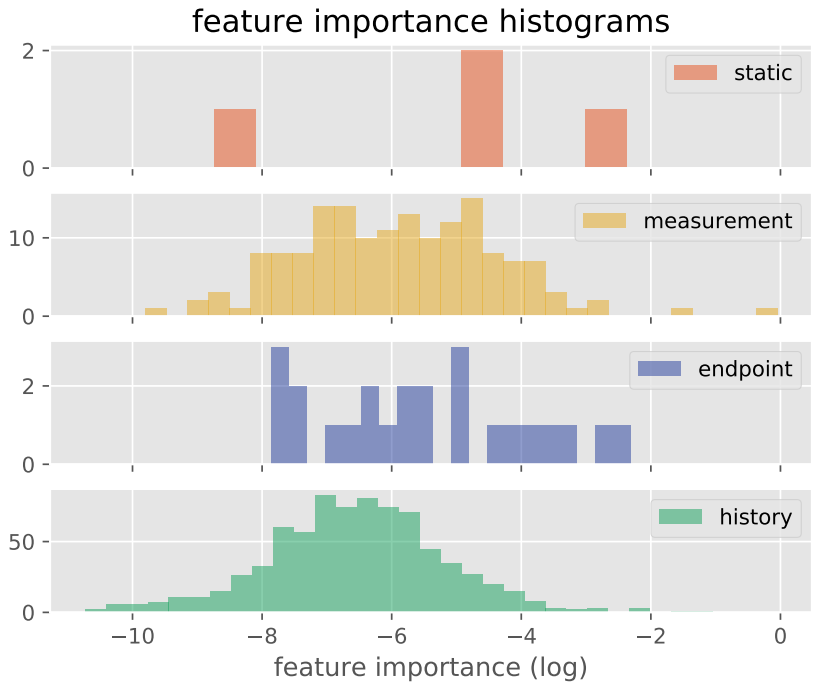


Figure 5.8: **Distribution of feature importances by class.** Histogram of (log) feature importances in the `reference` model computed using mean absolute SHAP value, broken down by feature class. Higher is more important.

Feature ablation

We can also assess the impact of different feature choices by running an ablation study, dropping feature classes from the `reference` model one by one. The resulting effect on the model's AUROC and AUPRC is shown in Figure 5.9. Perhaps unsurprisingly, the largest drop in performance comes from removing the most numerous feature classes (measurement and history), although as we see in Figure 5.8, these classes also contain many less important features, so the performance is likely driven by a small set of highly important features.

Figure 5.9: **Effect of removing different classes of features from the `reference` model.** Note the small variation in the y-axis. This indicates that no one feature class contributes substantially more than others. However, it also indicates that there is redundancy between feature classes.

Reduced model

Using the SHAP values described above, we can identify the top 20 *variables*. These variables, alongside their most important feature and the class it belongs to, are listed in Table 5.5. In the case of vasoactive drugs, they appear by virtue of their presence in the endpoint definition. We use these variables to define the `reduced` model.

Defining a model using a restricted set of variables means the model can more readily be applied ot other ICU settings, requiring less data extraction and

Table 5.5: **Top variables by mean absolute SHAP value.** These top variables, from the `reference` model are used to define the `reduced` model. A variable is ranked by its most important feature, which is shown in the second column. 'Vasoactive drugs' means *dobutamine*, *milrinone*, *levosimendan*, *theophyllin*. These are all potentially used to identify level 1 circulatory dysfunction, hence appear as a group.

variable	feature	class
Lactate	current value	measurement
Mean ABP	lowest values in last hour	history
Age	-	static
Time since admission	-	measurement
Vasoactive drugs*	fraction of last 24 hours with endpoint	endpoint
Systolic ABP	current imputed value	measurement
Heart rate	trend over last 12 hours	history
Respiratory rate	time since last measurement	measurement
RASS	current imputed value	measurement
Peak pressure	highest value in last 30 minutes	history
Weight	lowest value over last 72 hours	history
Creatine kinase	trend over last 48 hours	history
INR	time since last measurement	measurement
Glucose	trend over last 12 hours	history
C-reactive protein	trend over last 48 hours	history
Diastolic ABP	median value over last 12 hours	history
PEEP	time since last measurement	measurement

preparation. We demonstrate this in Section 5.5.6 by applying the `reduced` model to MIMIC. Moreover, since the variables listed in Table 5.5 are commonly measured, the `reduced` model can be applied in a greater diversity of ICUs.

5.5.3 Deterioration prediction

Having defined the `reference` and `reduced` models in Section 5.5.2, we can now compare their relative performance on the deterioration prediction task. The results are shown in Figure 5.10. We show the receiver-operating characteristic (ROC) curve and precision-recall curves for the model on the task of pre-

dicting deteriorations in the next 8 hours from stability (= level 0) to any level of circulatory failure. We show that the model achieves a high AUROC (0.93 for both models), and with a false alarm rate of 66%, approximately 70% of deteriorations are detected. As the prevalence of deteriorations is 3.8%, a random classifier would achieve a false alarm rate of 96.2% with arbitrarily high recall. Discussion with clinical collaborators suggested the precision point of 33% as providing a useful tradeoff between sensitivity and avoiding unnecessary false alarms, which contribute to alarm fatigue. Ultimately, determining the optimal threshold for the model would require a clinical study.

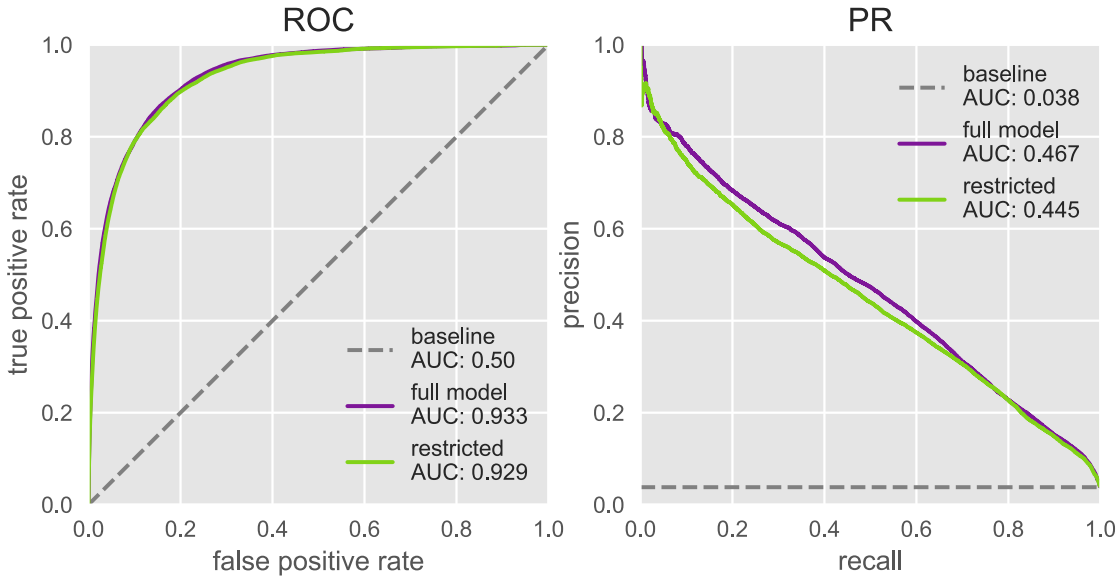


Figure 5.10: **Performance of the models in predicting circulatory system deterioration in the next 8 hours.** Receiver-operating characteristic and precision-recall curves for the `reference` and `reduced` models, which use 1000 features (on 198 variables) and 20 variables respectively. The baseline prevalence of positive events is 3.8% over all timepoints. Results in this experiment are reported on the fifth (most recent) temporal split. The precision-recall curve demonstrates the trade-off between false alarms and sensitivity. At a precision of 33%, 70% of deteriorations are detected.

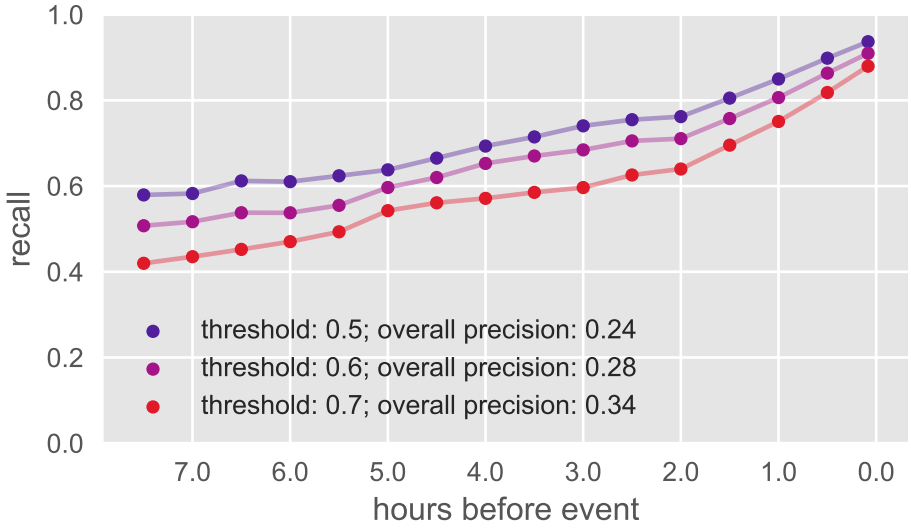


Figure 5.11: **Recall as a function of time before the event.** This is computed by binning the valid data in the 8 hours before the event into 30-minute intervals, and reporting the fraction of positive predictions in that interval (the label during this time is necessarily positive). Data must be valid in the sense that we make no predictions while patients are currently in an endpoint, and therefore cannot evaluate at these timepoints.

5.5.4 How much lead-time does the model provide?

While the model is trained to predict deteriorations in the next 8 hours, it seems likely that not every event is detected exactly 8 hours in advance. Ideally the model would provide sufficient notice for caregivers to take action. By construction of the prediction task, a true positive from the model means a deterioration will occur *at least* five minutes from now, but in the worst case, the model is only able to identify events occuring five minutes (and no later) from now. To assess this, we report the recall of the model, conditional on there being an event, as a function of time-to-deterioration.

This is shown in Figure 5.11. We see that while recall increases as time-to-event decreases, even 8 hours before the event, 50% of events are successfully

recalled at an overall precision of 28%. This indicates that the performance of the model is not driven by its performance in the moments before the event.

5.5.5 How well does the model generalise in time?

As described in Section 5.5.1, we split our data such that the validation and test sets contain the patients most recently admitted in the dataset. We additionally form five 'replicates' (see Figure 5.7) using sliding temporal windows, and a 'mixed' split, where the test (and validation) set is sampled independently from the entire time range (disjointly from the training set).

This setup allows us to ask several questions. In Table 5.6 we show the performance on each of the temporal test sets of models trained on the corresponding training set. We additionally include the 'mixed' split. We see three things here: firstly, there is no obvious trend in the generalisation error in different temporal splits, indicating that the level of domain shift (the difference between the training and test sets) is constant across the splits. This is a good sanity check, and enables us to treat these splits as 'replicates' for the purpose of training models. Secondly, we see that the prevalence in the test set varies slightly, making the AUPRC across splits not directly comparable. We address this in the next experiment. Finally, if we compare the performance on the 'mixed' split, we see that - unsurprisingly - using a test set sampled from the same temporal range as the training set results in superior performance. This underscores the fact that if we had applied the traditional train/test split without applying temporal stratification, we would have overestimated the true performance of the model.

Table 5.6: **Performance of model in different temporal splits.** We train a model on the training set of each temporal split and use its validation set for model selection, then report its test set performance. AUPRC in different splits is not directly comparable due to differences in positive label prevalence. We do not observe an obvious trend in AUROC, indicating that the generalisation error of the model is not time-dependent. Using the 'mixed' split, whose test set is selected from the same temporal distribution as the training set, we see that the temporal stratification approach is necessary to avoid overestimating the performance of the model. Values in brackets denote uncertainty (standard deviation) in the final digit over the temporal splits.

	late 2013	early 2014	late 2014	early 2015	late 2015	average	mixed
AUROC	0.9394	0.9301	0.9308	0.9346	0.9328	0.934(4)	0.9472
AUPRC	0.4825	0.4502	0.4858	0.4972	0.4696	0.48(2)	0.5317
prevalence	0.0426	0.0317	0.0351	0.0334	0.0377	0.036(4)	0.0384

We are then lead to ask the question - what if the test set were even *more* temporally removed? To what extent do we expect the performance of the model to degrade in time? To address this question, we re-purposed the temporal splits as follows: we *fix* a model, trained on the earliest temporal split. We then *test* that model on the test set of all splits in turn. This enables us to simulate the effect of the passage of time. The results are shown in Table 5.7. We can see from AUROC that there is no obvious trend of degradation in model performance. To attempt to control for the intrinsic 'difficulty' of the different test sets, we also show the performance (AUROC or AUPRC) of the model as a fraction of the performance of the model trained on the relevant training data set. That is to say, we compare the performance of the model trained on split 1 and tested on split 4 with the performance of a model trained on split 4 and tested on split 4. These results are the last two rows in Table 5.7 - note that the first column of Table 5.7 is identical to that of Table 5.6 since the late 2013 test set corresponds to the first split.

Once again, there is no obvious trend, however we do observe that the rel-

Table 5.7: **Performance of a fixed model as the test set varies.** We develop a model on the earliest split and test it on the test sets of subsequent splits, in order to assess if temporal distance results in a higher generalisation error. In the 'ratio' rows, we compare the performance in this setting with the results in Table 5.6, where the model would instead be retrained on more recent data. We don't see an obvious downward trend in the model's performance, indicating that dataset shift is not significant over a two-year period.

	late 2013	early 2014	late 2014	early 2015	late 2015
AUROC	0.9394	0.9300	0.9299	0.9343	0.9319
AUPRC	0.4825	0.4547	0.4814	0.4976	0.4588
prevalence	0.0426	0.0317	0.0351	0.0334	0.0377
AUROC ratio	1	0.9999	0.9990	0.9997	0.9990
AUPRC ratio	1	1.0010	0.9911	1.0009	0.9771

ative AUPRC on the farthest split (late 2015) is the lowest. This investigation indicates that if there will be a degradation in model performance due to domain shift as new data is collected, it will not become obviously apparent over a period of two years.

5.5.6 How well does the model generalise to a different ICU?

Given the `reduced` model, we can perform external validation by extracting the same features and labels from data from a different ICU. To enable reproducibility, we select the open-access dataset MIMIC-III [110] for this validation.

Processing MIMIC

In order to perform the comparison, we must identify and extract the equivalent variables from MIMIC. Given the top 20 most important variables (Table 5.5), we were able to identify equivalent variables in MIMIC. In the case of weight and glucose, unit conversions were required (lb $\rightarrow$ kg and mg/dl $\rightarrow$ mmol/l).

The MIMIC database contains data from two EHR systems - CareVue and MetaVision. The latter system came into effect in 2008, and we restrict our attention to data from this part of the database. Variables stored in MetaVision are indexed using a different set of unique identifiers, and MetaVision stores medication information in a different format to CareVue, so data processing is simplified greatly by applying this restriction. Since we also restrict the data from Bern to patients admitted after 2008, this may also increase similarity between the datasets.

Given the identified variables in MIMIC, we extract them from the relevant tables in the v1.4 export of MIMIC. Since many of the artefact removal steps in Section 5.3.1 are specific to the PDMS at Inselspital Bern, we apply a simpler form of artefact removal on MIMIC by only applying the physiological thresholding. We then pivot and merge tables and variables to produce data in the correct form to be processed by the rest of the pipeline developed for Bern. That is, we subsequently perform endpoint identification, missing data imputation, and label generation as described above, with minor modifications:

- **Imputation parameters**: part of our imputation pipeline requires calculating the sampling interval for each variable, as described in Section 5.3.4. At this point, we decided to *not* recompute these intervals on MIMIC, and instead apply the values computed from the Inselspital data. We did this for two reasons:

    1. If we consider these sampling intervals to reflect the *underlying* lengthscale for dynamics of that variable, it should not vary much between ICUs. *However*, the data in MIMIC appears to be artificially downsampled - MAP measurements are available approxi-

mately hourly, whereas in the Inselspital data, this variable is available every 2 minutes. We therefore prefer to use the Inselspital data, as we understand better how it has been recorded, and do not believe any measurements have been artificially downsampled (except inasmuch as is done already in the PDMS).

2. Furthermore, since we wish to evaluate our *model* on MIMIC, this model requires data processed in a particular way. The imputation parameters are implied to be contained in a fixed data pre-processing procedure which must be undertaken as a prerequisite to using the model. Therefore, recomputing these parameters would mean we are no longer evaluating the same model.

- **Time grid**: as mentioned above, the temporal resolution of MIMIC is different to that of Inselspital. Nonetheless, following the same arguments as above, we resample MIMIC to a five-minute grid, even if this introduces a large quantity of imputed data. The model we are evaluating was designed to operate on data at this resolution (regardless of imputation status), and we can therefore include this resampling procedure to also be part of the model.

During this processing, we observed that the drugs theophyllin and levosimendan are seemingly not used at Beth Israel Deaconess Medical Center. Instead, dopamine and phenylephrine are used, which also have vasoactive properties. These drugs are not used at Inselspital. We could therefore modify the endpoint definition for MIMIC to include dopamine and phenylephrine, while not requiring theophyllin or levosimendan. This would be equivalent to expanding the endpoint definition as described in section 5.2.1 to include

dopamine and phenylephrine, if available. This would result in 11.9% of patients having endpoints, as opposed to 10.6% using the original endpoint definition. To avoid making ICU-specific modifications to our endpoint definition, and since this change results in only a minor change in prevalence, we maintain the endpoint as already defined. However, in ICUs where vasoactive drugs use is more different, modifications of this kind will be necessary.

External performance

Having processed the MIMIC data, we consider two settings for evaluation:

- A model trained on Bern, but tested on MIMIC
- A model *trained* on MIMIC, and tested on MIMIC

The first case evaluates the 'pure' transfer problem: taking a model developed on Bern, and applying it directly (subject to the above data processing requirements) on data from another ICU. The second case rather evaluates our *methodology* for developing a model - including data processing, imputation, feature extraction, and model selection choices - and applies it to MIMIC. In Figure 5.12 we show ROC and PR curves for these settings. For reference, we also include the performance of the model trained on Bern and also evaluated on Bern (this is identical to the `reduced` result in Figure 5.10). Since events are slightly more rare in MIMIC (baseline prevalence 0.023, compare to 0.038 in Inselspital), to make precision performance comparable, we downsample negative examples in MIMIC until the baseline prevalence is equivalent.

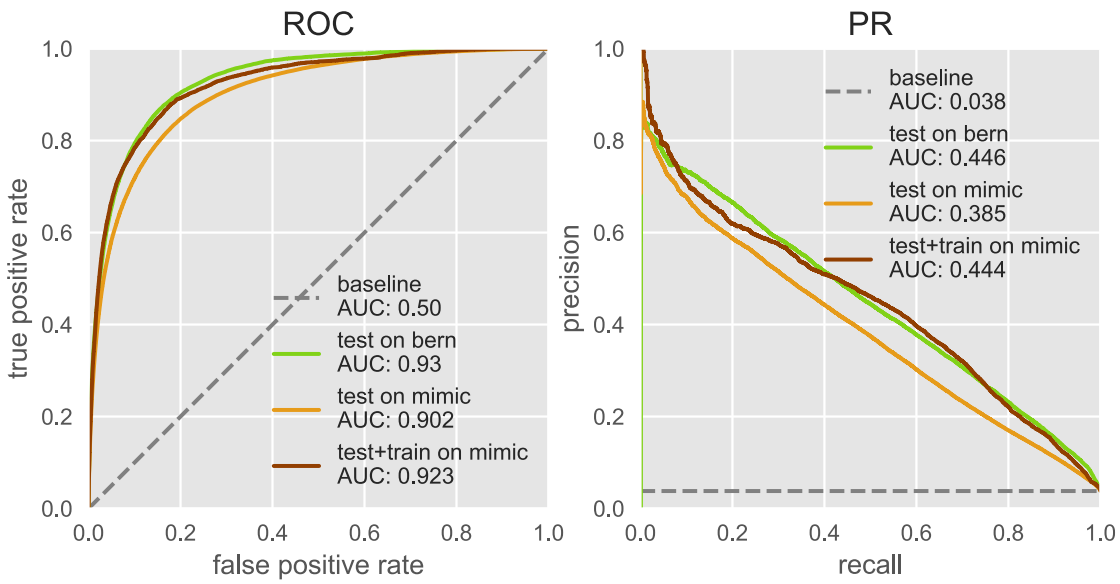


Figure 5.12: **Evaluation of the method on an open-access ICU dataset.** Comparison of the performance of the `reduced` model on Inselspital Bern (in green), on MIMIC (in orange), and the same model *trained* on MIMIC (in brown). We see that the highest performance is attained when the model is trained on data from the same domain as the test set, however the transferred model (in orange) still provides high performance, with AUROC = 0.902 and AUPRC = 0.385.

5.5.7 Event-based evaluation

For the model to excel in the previous analyses, it would need to consistently predict a deterioration for the entirety of the 8-hour period preceding the event (that is, the entirety of the period in which the label is positive). However, it may be sufficient to make a prediction at *any* point during this period for us to consider that that particular *event* has been correctly recalled. As shown in Figure 5.13, events are preceded by 71.4 alarms in 8 hours on average, which is likely to irritate clinicians and result in them disabling the system. This prompts another way of evaluating the model's performance, which we refer to as 'event-based'. Under this evaluation approach, we focus not on individually-labelled time-points, rather focusing on events and alarms. We can clearly define *recall* in this setting as the fraction of events which were correctly predicted, such

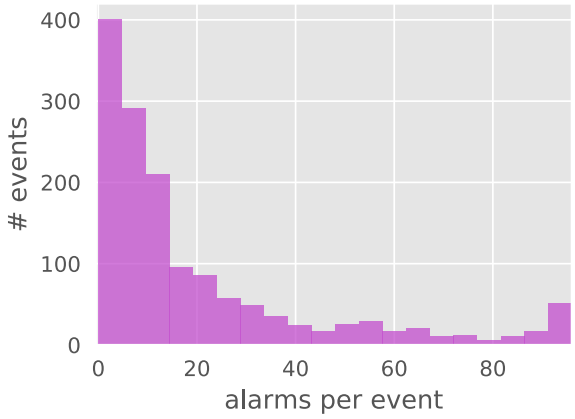


Figure 5.13: **Many alarms are triggered for each event.** The maximum number of alarms is 96 (=12*8), although the period before an event in which alarms could be triggered can be shorter than 8 hours. Overall, 71.4 alarms are triggered on average for each event. The threshold used here is 0.6.

that there was any positive alarm in the 8 hours preceding the event. To define precision, we return to focusing on alarms, asking what fraction of all *alarms* are true, in that an event occurs within 8 hours.

It is possible for two events to occupy a single 8-hour window, such that an alarm triggered in that window *could* arguably be said to be predicting either of these events. To avoid double-counting, we take the conservative approach to only allow an alarm to contribute to the identification of a *single* event, which would be the first event after the alarm. This means that an alarm followed by two events within 8 hours would have a precision of 100%, but a recall of 50%.

In Figure 5.14 we show the precision-recall curve under this evaluation, compared to the performance under the typical evaluation (as in Figure 5.10). We use the `reference` model in both settings. We see that defining recall using events rather than alarms produces a higher precision for the same recall. This indicates that if it suffices to produce a *single* alarm to predict an event, a higher precision can be achieved by increasing the alarm threshold accordingly.

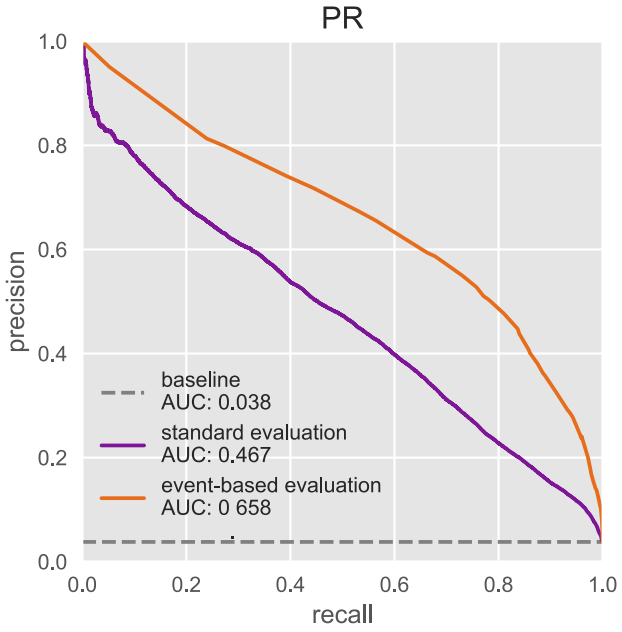


Figure 5.14: **Impact of calculating recall in terms of events.** Defining recall in terms of events rather than alarms improves the precision-recall trade-off relative to using the standard evaluation. This indicates that if whole events are more relevant than individual labels (of which there will be many, per event), a higher precision can be achieved. With a precision of 50%, 80% of events can be predicted.

Alarm silencing

To directly address the (potential) issue of bothersome alarms, we implement a modification on top of the classifier which *silences* any alarms for a set duration after an alarm has been triggered. This is intended to emulate a possible mode of operation for clinicians, where they intentionally disable the system while they tend to the patient. Alarm silencing effectively turns some false (or true) positives into true (or false) negatives. The extent to which this harms the model depends on how these events are clustered in time. If, for example, false positives occur in groups, then silencing will appear to benefit the model, as more erroneous alarms are silenced. We show in Figure 5.15 (left) how silencing

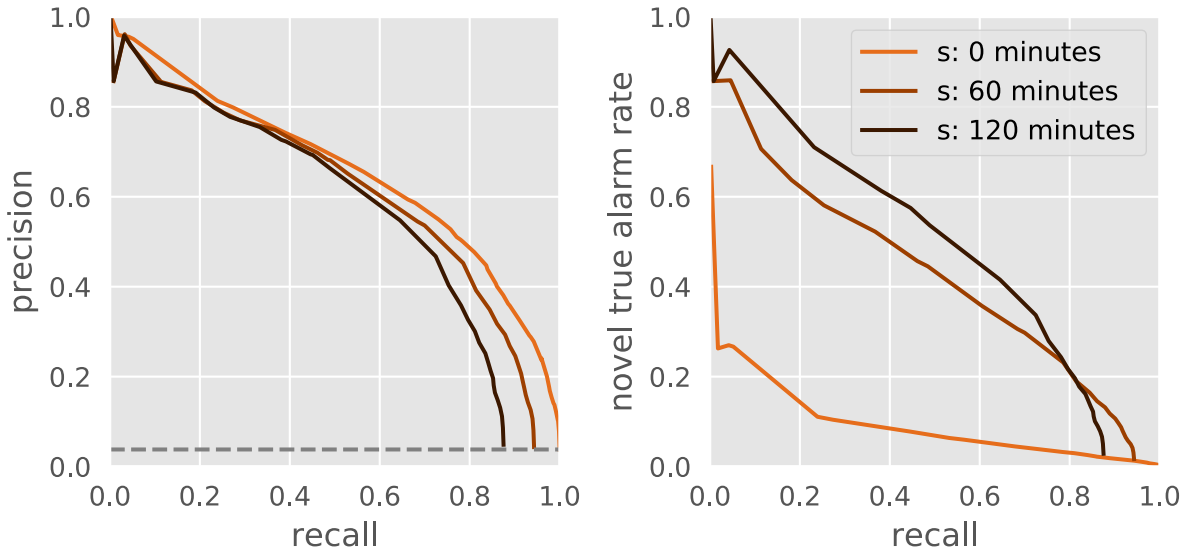


Figure 5.15: **Silencing alarms harms recall, but increases alarm novelty.** We show precision versus recall (computed in terms of events) for an alarm system with three silencing levels (s): none, 60 minutes, and 120 minutes. The system is silenced for s minutes after every alarm (independent of whether the alarm is true or not, which is unknown at the time of silencing). On the left, we see that silencing harms the maximum attainable recall, due to events which happen during the silencing period. On the right, we report what fraction of alarms are *novel* (or interesting), which means they are true alarms and predicting an event which has not already been predicted. In this case, we see that silencing greatly improves this novelty rate, confirming the observation that many true alarms are simply repeatedly alerting the same event.

times of 0 (default), 60, and 120 minutes impact the resulting precision-recall curve of the model, where recall is now defined in the event-based way. We see that the theoretical maximum recall is decreased with silencing, indicating that some events occur so soon after other (correctly predicted) events that silencing makes them 'impossible' to predict. To account for this, the alarm should be unsilenced when the patient enters an event, either automatically or manually.

We note that the number of alarms per event drops to 3.2 using 120 minutes of silencing, from 71.4 without any silencing. To be comparable with Figure 5.13, where the overall recall is 0.75 with threshold 0.6, we use the threshold of 0.73 to achieve comparable recall. This results in 3.2 alarms per event, with a precision

of 55% (compared to 30% without silencing).

In the right panel of Figure 5.15, we do not report precision against recall. Here, we ask what fraction of alarms are both true and *novel* - that is, the alarm is for an event which has not already been predicted. This could be considered a measure of what fraction of alarms are 'interesting' (where false alarms cannot be interesting). This is a setting under which the default model (with no silencing) is bound to fail - and does - since this model was trained to ideally produce 96 (=12*8) alarms *per* event. In this case, silencing clearly helps increase the fraction of novel alarms, although the recall still suffers from a hard upper bound. These results indicate how choosing the optimal system requires carefully considering how to evaluate it, and that it will be necessary to understand how clinicians behave towards nuisance alarms.

5.6 Conclusions and future work

This chapter describes an applied machine learning solution to a real clinical problem. Going from routinely-collected data stored in an ICU patient database management system, in close collaboration with clinicians we went through data processing, task and model specification and training, and evaluation. This kind of secondary use of electronic health data is challenging due to the unreliable nature of the data, necessitating a substantial effort in data preparation and understanding. Although the model demonstrates high performance in the prediction task, the work in this chapter defines just the first necessary steps towards building an early-warning system which can be used in practice. Realisation of such a system further needs:

- Development of a prototype system which can run on hospital infrastructure - we have developed a system which uses a static view of the PDMS: data processing will need to be re-engineered to operate in real time on continuously-measured data.
- Communication of the model predictions to clinicians - the model produces a score between 0 and 1 every five minutes, but we have thus far only evaluated the *alarms* triggered by this score exceeding a particular threshold. Understanding the properties of the score itself (its trends, its noise) is necessary before this score is provided as a diagnostic aid.
- Ultimately, such a tool must be tested in a randomised clinical trial to assess its impact in terms of patient outcomes and clinician burden.

Beyond deployment, there remain many research questions, such as:

- Development of a 'clinical baseline' - how well do nurses or doctors perform at the same task? Comparisons between algorithms and human practitioners is always fraught with danger of misinterpretion [169], so defining a fair comparison in terms of task and how to present that task is a non-trivial, but perhaps illuminating endeavour
- Initial experiments indicated that predicting *further* deteriorations in terms of circulatory function is a significantly more difficult task. This should be revisited in light of the success in the easier setting. It may be that the difference between levels 2 and 3 (see Table 5.1b) is simply too arbitrary, and merging these results in more coherent labels for the classifier. It may be that learning all prediction problems simultaneously allows for information transfer between tasks, but it necessitates careful definition of labels.

It may be that *pretraining* on one task is enhances performance in tasks (such as predicting $2 \rightarrow 3$) with insufficient training data.

- Initial experiments also indicate that LSTM-based models do not outperform lightGBM in this task. While this is not surprising, it is likely that a sufficiently well-engineered deep architecture would be more successful, and could identify aspects of the data which were overlooked during feature engineering.
- A large degree of effort was expended on data preprocessing, including correcting human errors during data entry, removing measurement artefacts, merging semantically similar variables, and missing data imputation. This makes developing such a system costly in terms of required time and expertise. Designing systems to automate or help in theses tasks would benefit researchers as well as informatics practitioners. Some of these already exist, for example the ACHILLES system[3] or cleanEHR[4], but these workflows nonetheless require further development.
- The external validation on MIMIC in Section 5.5.6 required manually matching variables, and poses a barrier to the testing of this model in other hospitals. Translating the Inselspital PDMS dataset into a common data model such as OMOP[5] with a standardised terminology would enable the system developed in this chapter to be more readily adopted elsewhere.
- In this chapter we looked at the circulatory system. Similar endeavours should be undertaken to address other organ systems, such as the respiratory and renal systems. It would then be possible to jointly model these organ systems to produce an overall severity of illness score, akin to the

[3] https://www.ohdsi.org/analytic-tools/achilles-for-data-characterization/
[4] https://github.com/ropensci/cleanEHR
[5] https://github.com/OHDSI/CommonDataModel

SOFA score [220], but better exploiting the temporal nature of the data.

Moreover, given the dataset described in this chapter, many clinical questions can be asked, like:

- To what extent can you predict lactate alone? Can improved lactate prediction result in a more accurate early warning system?
- Can you identify patients for which certain types of treatments are not beneficial, such as fluid boluses or intubation?
- To what extent does time of day impact care delivery and treatment patterns? Time of day was not used as a feature in the system in this chapter, but it may be informative. Do caregivers make more mistakes towards the end of their shift, do they avoid initiating time-consuming procedures, or do they behave more conservatively?

For data of this size and complexity, the limiting factor is choosing the right question.

CHAPTER 6

CONCLUSION

> This isn't life in the fast lane, it's life in the oncoming traffic.
>
> *Terry Pratchett*

With an increasing drive towards digitisation of health - in hospitals, in the hands of patients - medicine is at the frontier of becoming a data-driven discipline. To fully benefit from these changes, we need to render this collected data into actionable clinical knowledge. Concurrently, machine learning research is making tremendous strides in areas such as speech and image recognition, natural language processing, and recommender systems. These advances have come in part due to the ever-increasing power and availability of computational resources, algorithmic and theoretical advances, and a global explosion of interest in the field. However, healthcare has yet to enjoy a machine learning revolution. This may be in part due to the sheer complexity of medical data, and the challenges of working with it, practically and ethically. The work in this dissertation aimed to address some of these challenges, and to help to bring data-driven, artificially intelligent healthcare one step closer to reality.

In particular, this dissertation has focused on problems in natural language processing for medicine, and time series analysis, generation and classification. We have developed methods to learn semantically meaningful representations of medical concepts, leveraging the distributional hypothesis from linguistics. We have seen how the correspondence between Lie groups and Lie algebras can be exploited in machine learning to define a full parametrisation of unitary

matrices to be used in recurrent neural networks. We have demonstrated that the framework of generative adversarial training can be used to generate useful synthetic multivariate medical time series, and that such synthetic data can be analysed for its utility and privacy purposes. Finally, we have glimpsed how the hospital of tomorrow may exploit machine learning to build predictive systems capable of processing data on a scale inconceivable to human doctors to identify patients at risk of imminent circulatory deterioration.

Many open questions remain. For example,

- How can we merge different modalities of data to learn joint representations of patients, taking into account time-varying physiology, time-invariant demographic attributes, radiology, histopathology, doctors' notes, and self-reported outcome measures? We have focused here on particular aspects of this data, but the full picture demands more. To further our characterisation of patient state, we must also look beyond what is currently recorded in routine use and ask how machine learning enables the use of new data streams.
- The characterisation of patient state is only the first step. Such characterisations must be used. We have seen how machine learning enables early detection of imminent physiological deterioration, but how far can this go? To what extent can we pick up on the signs of ill health, and how early? Circulatory deterioration is one problem of many. Is there shared structure between medical prediction problems, and can we transfer successes in one problem to others?
- The dream of machine learning in healthcare does not end with prediction. We would like to create systems which can also advise clinicians on

how to act. Such expert systems are not a new concept, however making such systems accurate, reliable, and interpretable is an ongoing avenue of research. The alarm system described in Chapter Five does not make recommendations for specific action, but it is easy to imagine future systems which provide alerts as well as suggestions for how, perhaps, to best prevent the impending deterioration. Thus, such systems may attempt to identify optimal *sequences* of actions, placing it in the subfield of reinforcement learning, or calling for techniques from the study of multi-armed bandits.

Throughout all of this, we must remember that healthcare is an endeavour ultimately delivered by people to other people. The technological solutions we propose must be grounded in an understanding of how they will be used, and the impact they will have on the lives of real people. Medicine is vastly complex, and the system we use to deliver it is growing ever more complex. The march of technological progress may drive this complexity, but it can also save us from it. Building the right tools, for the right problems, will transform a deluge of data into life-saving knowledge.

BIBLIOGRAPHY

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] Jerome Aboab, Leo Anthony Celi, Peter A. Charlton, Mengling Feng, Mohammad Ghassemi, Dominic C Marshall, Louis Mayaud, Tristan Naumann, N Mccague, Kenneth E Paik, Tom J. Pollard, Matthieu Resche-Rigon, Justin D Salciccioli, and David J. Stone. A 'datathon' model to support cross-disciplinary collaboration. *Science Translational Medicine*, 8: 333ps8–333ps8, 2016.

[3] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2009.

[4] Ahmed M. Alaa, Scott Hu, and Mihaela van der Schaar. Learning from clinical judgments: Semi-markov-modulated marked hawkes processes for risk prognosis. In *ICML*, 2017.

[5] Google Code Archive. word2vec. `https://code.google.com/archive/p/word2vec/`, 2013. URL `https://code.google.com/archive/p/word2vec/`.

[6] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *International Conference on Learning Representations (ICLR) - Workshop track*, 2016.

[7] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[8] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *STOC*, 2017.

[9] Karla L. Caballero Barajas and Ram Akella. Dynamically modeling patient's health state from electronic medical records: A time series approach. In *KDD*, 2015.

[10] Juliana Barr, Gilles L. Fraser, Kathleen A Puntillo, Eugene Wesley Ely, Céline Gélinas, Joseph F. Dasta, Judy E. Davidson, John W. Devlin, John Paul Kress, Aaron M. Joffe, Douglas B. Coursin, Daniel L. Herr, Avery Tung, Bryce R. H. Robinson, Dorrie K Fontaine, Michael A. E. Ramsay, Richard R. Riker, Curtis N. Sessler, Brenda Truman Pun, Yoanna Skrobik, and Roman J. Jaeschke. Clinical practice guidelines for the management of pain, agitation, and delirium in adult patients in the intensive care unit. *Critical care medicine*, 41 1:263–306, 2013.

[11] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[12] Andrew L. Beam, Benjamin Kompa, Inbar Fried, Nathan P. Palmer, Xu Shi, Tianxi Cai, and Isaac S. Kohane. Clinical concept embeddings learned from massive sources of medical data. *CoRR*, abs/1804.01486, 2018.

[13] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, and Casey S.

Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv*, 2017. doi: 10.1101/159756. URL https://www.biorxiv.org/content/early/2017/07/05/159756.

[14] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[15] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[16] Sakyajit Bhattacharya, Vaibhav Rajan, and Harsh Shrivastava. Icu mortality prediction: A classification algorithm for imbalanced datasets. In *AAAI*, 2017.

[17] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *ICLR*, 2018.

[18] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 546–556. Association for Computational Linguistics, 2012.

[19] Google Official Blog. Introducing the knowledge graph: things, not strings. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, 2012. URL https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html.

[20] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, 32:D267–D270, 2004.

[21] Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. Weakly-supervised alignment of video with text. In *Proceedings of the IEEE international conference on computer vision*, pages 4462–4470, 2015.

[22] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, 2008.

[23] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[24] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135, 2012.

[25] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.

[26] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[27] Lars Borin, Markus Forsberg, and Lennart Lönngren. Saldo: a touch of yin to wordnet's yang. *Language Resources and Evaluation*, 47(4):1191–1211, Dec 2013. ISSN 1574-0218. doi: 10.1007/s10579-013-9233-4. URL https://doi.org/10.1007/s10579-013-9233-4.

[28] Ali Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

[29] Wacha Bounliphone, Eugene Belilovsky, Matthew Blaschko, Ioannis Antonoglou, and Arthur Gretton. A test of relative similarity for model selection in generative models. *ICLR*, 2016.

[30] Kate Bray, Ian Wren, Andrea R. Baldwin, Una St Ledger, Vanessa Gibson, Sheila Goodman, and Dominic Walsh. Standards for nurse staffing in critical care units determined by: The british association of critical care nurses, the critical care networks national nurse leads, royal college of nursing critical care and in-flight forum. *Nursing in critical care*, 15 3:109–11, 2010.

[31] Jacob S. Calvert, Daniel A. Price, Uli K. Chettipally, Christopher W. Barton, Mitchell D. Feldman, Jana L. Hoffman, Melissa Jay, and Ritankar Das. A computational approach to early sepsis detection. *Computers in biology and medicine*, 74:69–73, 2016.

[32] Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41 – 75, 1997. ISSN 08856125. doi: 10.1023/A:1007379606734.

[33] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[34] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David A Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. In *Scientific Reports*, 2018.

[35] Huanhuan Chen, Fengzhen Tang, Peter Tiňo, and Xin Yao. Model-based kernel for efficient time series analysis. In *KDD*, 2013.

[36] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180, 2016.

[37] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Retain: Interpretable predictive model in healthcare using reverse time attention mechanism. In *NIPS*, 2016.

[38] Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *JMLR workshop and conference proceedings*, 56:301–318, 2016.

[39] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine Learning for Healthcare Conference*, pages 286–305, 2017.

[40] François Chollet et al. Keras. `https://keras.io`, 2015.

[41] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[42] Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. *International Conference on Machine Learning (ICML)*, 2014.

[43] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. Torch: a modular machine learning software library. Technical report, Idiap, 2002.

[44] Michael Cook and Simon Colton. Ludus ex machina: Building a 3d game designer that competes alongside humans. In *ICCC*, 2014.

[45] Chris Culnane, Benjamin IP Rubinstein, and Vanessa Teague. Health data in an open world. *arXiv preprint arXiv:1712.05627*, 2017.

[46] Marco Cuturi and Arnaud Doucet. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*, 2011.

[47] Etienne Decencière, Xiwei Zhang, Guy Cazuguel, Bruno Lay, Béatrice Cochener, Caroline Trone, Philippe Gain, Richard Ordonez, Pascale Massin, Ali Erginay, et al. Feedback on a publicly distributed image database: the messidor database. *Image Analysis & Stereology*, 33(3):231–234, 2014.

[48] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[49] Kerstin Denecke and Yihan Deng. Sentiment analysis in medical settings: New opportunities and challenges. *Artificial intelligence in medicine*, 64(1): 17–27, 2015.

[50] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and*

Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. Ieee, 2009.

[51] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[52] Cicero dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[53] DC Dowson and BV Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

[54] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9 (3–4):211–407, 2014.

[55] Cristóbal Esteban, Oliver Staeck, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 93–101, 2016.

[56] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

[57] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[58] Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. Transition-based knowledge graph embedding with relational mapping

properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, pages 328–337, 2014.

[59] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics, 2015.

[60] Jeffrey De Fauw, Joseph R. Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, George van den Driessche, Balaji Lakshminarayanan, Clemens Meyer, Faith Mackinder, Simon Bouton, Kareem W. Ayoub, Reena Chopra, Dominic King, Alan Karthikesalingam, Cían O Hughes, Rosalind A Raine, Julian C. Hughes, Dawn A. Sim, Catherine Egan, Adnan Tufail, Hugh Montgomery, Demis Hassabis, Geraint Rees, Trevor Back, Peng T. Khaw, Mustafa Suleyman, Julien Cornebise, Pearse A. Keane, and Olaf Ronneberger. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*, 24:1342–1350, 2018.

[61] Christiane Fellbaum. Wordnet : An electronic lexical database. 1998.

[62] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31:59–79, 2010.

[63] Samuel G. Finlayson, Paea LePendu, and Nigam Haresh Shah. Building

the graph of medicine from millions of clinical narratives. In *Scientific data*, 2014.

[64] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[65] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security*, 2015.

[66] Daniel Fried and Kevin Duh. Incorporating both distributional and relational semantics in word representations. In *Workshop Contribution at the International Conference on Learning Representations (ICLR), 2015*, 2014.

[67] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[68] Haiyan Gao, Ann Mcdonnell, David A Harrison, Tracey Moore, Sheila K Adam, Kathleen Daly, Lisa Esmonde, David R. Goldhill, Gareth J. G. Parry, Arash Rashidian, Christian Peter Subbe, and Sheila Harvey. Systematic review and evaluation of physiological track and trigger warning systems for identifying at-risk patients on the ward. *Intensive Care Medicine*, 33:667–679, 2007.

[69] Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[70] Marzyeh Ghassemi, Tristan Naumann, Finale Doshi-Velez, Nicole Brimmer, Rohit Joshi, Anna Rumshisky, and Peter Szolovits. Unfolding physiological state: mortality modelling in intensive care units. *KDD : proceed-*

ings. International Conference on Knowledge Discovery & Data Mining, 2014: 75–84, 2014.

[71] Shameek Ghosh, Jinyan Li, Longbing Cao, and Kotagiri Ramamohanarao. Septic shock prediction for icu patients via coupled hmm walking on sequential contrast patterns. *Journal of biomedical informatics*, 66:19–31, 2017.

[72] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[73] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[74] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[75] Ben Graham. Kaggle diabetic retinopathy detection competition report. *University of Warwick*, 2015.

[76] Rolando Claure-Del Granado and Ravindra L Mehta. Fluid overload in the icu: evaluation and management. In *BMC nephrology*, 2016.

[77] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf,

and Alex J Smola. A kernel method for the two-sample-problem. In *NIPS*, pages 513–520, 2006.

[78] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[79] Paulina Grnarova, Florian Schmidt, Stephanie L. Hyland, and Carsten Eickhoff. Neural document embeddings for intensive care patient mortality prediction. *CoRR*, abs/1612.00467, 2016.

[80] Paulina Grnarova, Florian Schmidt, Stephanie L Hyland, and Carsten Eickhoff. Neural document embeddings for intensive care patient mortality prediction. *arXiv preprint arXiv:1612.00467*, 2016.

[81] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pages 5769–5779, 2017.

[82] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[83] David A Harrison, Gareth J. G. Parry, James R. Carpenter, Alasdair Short, and Kathy Rowan. A new risk prediction model for critical care: the intensive care national audit & research centre (icnarc) model. *Critical care medicine*, 35 4:1091–8, 2007.

[84] David A Harrison, Nazir I Lone, Catriona Haddow, Moranne MacGillivray, Angela A. Khan, Brian Cook, and Kathryn M. Rowan. External validation of the intensive care national audit & research centre (icnarc) risk prediction model in critical care units in scotland. In *BMC anesthesiology*, 2014.

[85] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *CoRR*, abs/1703.07771, 2017.

[86] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. 2018.

[87] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Learning rotations with little regret. *Machine Learning*, pages 1–20, 2016. ISSN 1573-0565. doi: 10.1007/s10994-016-5548-x.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[89] Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *ICML*, 2018.

[90] Mikael Henaff, Arthur Szlam, and Yann LeCun. Orthogonal rnns and long-memory tasks. *arXiv preprint arXiv:1602.06662*, 2016.

[91] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560, 2018.

[92] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[93] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[94] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, 1991.

[95] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[96] Jeremy Howard. Now anyone can train imagenet in 18 minutes. `http://www.fast.ai/2018/08/10/fastai-diu-imagenet/`, 2018. URL `http://www.fast.ai/2018/08/10/fastai-diu-imagenet/`.

[97] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[98] Xun Huang, Yixuan Li, Omid Poursaeed, John E. Hopcroft, and Serge J. Belongie. Stacked generative adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1875, 2017.

[99] Vijay Huddar, Bapu Koundinya Desiraju, Vaibhav Rajan, Sakyajit Bhattacharya, Shourya Roy, and Chandan K. Reddy. Predicting complications in critical care using heterogeneous clinical data. *IEEE Access*, 4:7988–8001, 2016.

[100] Stephanie L. Hyland and Gunnar Rätsch. Learning unitary operators with help from u(n). In *AAAI*, 2017.

[101] Stephanie L. Hyland, Theofanis Karaletsos, and Gunnar Rätsch. Knowledge transfer with medical language embeddings. *NIPS Workshop on Machine Learning for Healthcare*, abs/1602.03551, 2016.

[102] Stephanie L Hyland, Theofanis Karaletsos, and Gunnar Rätsch. A generative model of words and relationships from multiple sources. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[103] Stephanie L Hyland, Cristóbal Esteban, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[104] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[105] Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. Ontologically grounded multi-sense representation learning for semantic vector space models. In *HLT-NAACL*, 2015.

[106] Robert I Jennrich and Peter B Bright. Fitting systems of linear differential equations using computer generated exact derivatives. *Technometrics*, 18 (4):385–392, 1976.

[107] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

[108] Richard Johansson and Luis Nieto Piña. Embedding a semantic network in a word space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pages 1428–1433. Association for Compu-

tational Linguistics, May–June 2015. URL http://www.aclweb.org/anthology/N15-1164.

[109] Alistair E. W. Johnson, Tom J. Pollard, and Roger G. Mark. Reproducibility in critical care: a mortality prediction case study. In Finale Doshi-Velez, Jim Fackler, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 361–376, Boston, Massachusetts, 18–19 Aug 2017. PMLR. URL http://proceedings.mlr.press/v68/johnson17a.html.

[110] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[111] Siddhartha Jonnalagadda, Trevor Cohen, Stephen Wu, and Graciela Gonzalez. Enhancing clinical concept extraction with distributional semantics. *Journal of biomedical informatics*, 45(1):129–140, 2012.

[112] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Measuring the quality of synthetic data for use in competitions. *arXiv preprint arXiv:1806.11345*, 2018.

[113] JD Kalbfleisch and Jerald F Lawless. The analysis of panel data under a markov assumption. *Journal of the American Statistical Association*, 80(392): 863–871, 1985.

[114] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong

Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

[115] Joseph B Keller. Closest unitary, orthogonal and hermitian operators to a given operator. *Mathematics Magazine*, 48(4):192–197, 1975.

[116] Fiona E. Kelly, Kevin J Fong, Nicholas Hirsch, and Jerry P Nolan. Intensive care medicine is 60 years old: the history and future of the intensive care unit. *Clinical medicine*, 14 4:376–9, 2014.

[117] Curtis E. Kennedy and James P. Turley. Time series analysis as input for clinical predictive modeling: Modeling cardiac arrest in a pediatric icu. In *Theoretical Biology and Medical Modelling*, 2010.

[118] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C. Rindflesch. Semmeddb: a pubmed-scale repository of biomedical semantic predications. *Bioinformatics*, 28(23):3158–3160, 2012. doi: 10.1093/bioinformatics/bts591. URL `http://bioinformatics.oxfordjournals.org/content/28/23/3158.abstract`.

[119] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[120] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[121] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[122] J R Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel

Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[123] William A. Knaus, Douglas P. Wagner, Elizabeth A. Draper, Jack E. Zimmerman, Martin Bergner, P. G. Bastos, Carl A. Sirio, Deirdre J. Murphy, T Lotring, and Anne Damiano. The apache iii prognostic system. risk prediction of hospital mortality for critically ill hospitalized adults. *Chest*, 100 6:1619–36, 1991.

[124] Jan Koutník, Klaus Greff, Faustino J. Gomez, and Jürgen Schmidhuber. A clockwork rnn. In *ICML*, 2014.

[125] Rahul G. Krishnan, Uri Shalit, and David A Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, 2017.

[126] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[127] Denis Krompaß, Cristóbal Esteban, Volker Tresp, Martin Sedlmayr, and Thomas Ganslandt. Exploiting latent embeddings of nominal clinical data for predicting hospital readmission. *KI - Künstliche Intelligenz*, 29(2):153–159, 2014. ISSN 1610-1987. doi: 10.1007/s13218-014-0344-x. URL `http://dx.doi.org/10.1007/s13218-014-0344-x`.

[128] David Krueger and Roland Memisevic. Regularizing rnns by stabilizing activations. *International Conference on Learning Representations (ICLR)*, 2016.

[129] Pat Langley. The changing science of machine learning. *Machine Learning*, 82(3):275–279, 2011.

[130] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[131] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[132] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[133] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195, 2015.

[134] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, 2014.

[135] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[136] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, 2015.

[137] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wet-

zel. Learning to diagnose with lstm recurrent neural networks. *CoRR*, abs/1511.03677, 2015.

[138] James Robert Lloyd and Zoubin Ghahramani. Statistical model criticism using kernel two sample tests. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 829–837, Cambridge, MA, USA, 2015. MIT Press.

[139] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *CoRR*, abs/1711.10337, 2017.

[140] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[141] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *CoRR*, abs/1802.03888, 2018.

[142] Yuan Luo, Yu Xin, Rohit Joshi, Leo A. Celi, and Peter Szolovits. Predicting icu mortality risk by grouping temporal trends from a multivariate panel of physiologic measurements. In *AAAI*, 2016.

[143] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, 2013.

[144] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *KDD*, 2017.

[145] Alexa T. McCray. An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics*, 4:80 – 84, 2003.

[146] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine learning (ICML)*, 2018.

[147] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *NIPS*, 2017.

[148] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *ICLR*, abs/1611.02163, 2017.

[149] Alexander Meyer, Dina Zverinski, Boris Pfahringer, Jörg Kempfert, Titus Kühne, Simon H. Sündermann, Christof Stamm, Thomas Hofmann, Volkmar Falk, and Carsten Eickhoff. Machine learning for real-time prediction of complications in critical care: a retrospective study. *The Lancet. Respiratory medicine*, 2018.

[150] Zakaria Mhammedi, Andrew D. Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *ICML*, 2017.

[151] Xu Miao and Rajesh PN Rao. Learning the lie groups of visual invariance. *Neural computation*, 19(10):2665–2693, 2007.

[152] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[153] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their composition-

ality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[154] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *International Conference on Learning Representations (ICLR) - Workshop track*, 2015.

[155] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

[156] Randolph A Miller, Harry E Pople Jr, and Jack D Myers. Internist-i, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476, 1982.

[157] Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T. Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. In *Scientific reports*, 2016.

[158] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 6 November 2014.

[159] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[160] SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. Distributional semantics resources for biomedical text processing. pages 39–43, 2013.

[161] Linda J. Moniz, Anna L. Buczak, Lang M. Hung, Steven Babin, Michael Dorko, and Joseph M Lombardo. Construction and validation of synthetic electronic medical records. In *Online journal of public health informatics*, 2009.

[162] Travis J Moss, Douglas E. Lake, James Forrest Calland, Kyle B. Enfield, John B. Delos, Karen D. Fairchild, and J. Randall Moorman. Signatures of subacute potentially catastrophic illness in the icu: Model development and validation. *Critical care medicine*, 44 9:1639–48, 2016.

[163] Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*, 2016.

[164] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. IEEE, 2008.

[165] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[166] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 2017.

[167] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[168] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104:11–33, 2016.

[169] Luke Oakden-Rayner. Do machines actually beat doctors? `https://lukeoakdenrayner.wordpress.com/2016/11/27/do-computers-already-outperform-doctors/`, 2016.

[170] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.

[171] Catherine Olsson, Surya Bhupatiraju, Tom B. Brown, Augustus Odena, and Ian J. Goodfellow. Skill rating for generative models. *CoRR*, abs/1808.04888, 2018.

[172] OpenAI. Openai five. `https://blog.openai.com/openai-five/`. URL `https://blog.openai.com/openai-five/`.

[173] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition. *Linguistic Data Consortium*, 2011.

[174] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

[175] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[176] Trang Pham, Truyen Tran, Dinh Q. Phung, and Svetha Venkatesh. Deepcare: A deep dynamic memory model for predictive medicine. In *PAKDD*, 2016.

[177] Trang Pham, Truyen Tran, Dinh Q. Phung, and Svetha Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of biomedical informatics*, 69:218–229, 2017.

[178] Michael R. Pinsky, Gilles Clermont, and Marilyn Hravnak. Predicting cardiorespiratory instability. In *Critical care*, 2016.

[179] Tom J. Pollard, Alistair E. W. Johnson, Jesse D. Raffa, Leo A. Celi, Roger G. Mark, and Omar Badawi. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific Data*, 5:180178, September 2018. doi: 10.1038/sdata.2018.178. URL `https://www.nature.com/articles/sdata2018178`.

[180] Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *arXiv preprint arXiv:1804.06323*, 2018.

[181] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[182] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M. Dai, Nissan Hajaj, Peter J. Liu, Xiaobing Liu, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Gavin E. Duggan, Gerardo Flores, Michaela Hardt, Jamie Irvine, Quoc V. Le, Kurt Litsch, Jake Marcus, Alexander Mossin, Justin Tan-

suwan, De Wang, James Wexler, Jimbo Wilson, Dana Ludwig, Samuel L. Volchenboum, Katherine Chou, Michael Pearson, Srinivasan Madabushi, Nigam Haresh Shah, Atul J. Butte, Michael Howell, Claire Cui, Gregory S. Corrado, and Jeff Dean. Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1:1–10, 2018.

[183] Luke Oakden Rayner. Exploring the chestxray14 dataset: problems. https://lukeoakdenrayner.wordpress.com/2017/12/18/the-chestxray14-dataset-problems/, 2017. URL https://lukeoakdenrayner.wordpress.com/2017/12/18/the-chestxray14-dataset-problems/.

[184] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[185] Thomas C Rindflesch and Marcelo Fiszman. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36(6):462 – 477, 2003. ISSN 1532-0464. doi: http://dx.doi.org/10.1016/j.jbi.2003.11.003. URL http://www.sciencedirect.com/science/article/pii/S1532046403001175. Unified Medical Language System.

[186] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[187] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual

Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[188] Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–53, 2008.

[189] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pages 2234–2242, 2016.

[190] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations (ICLR)*, 2014.

[191] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, 1997.

[192] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.

[193] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[194] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[195] Edward H Shortliffe and Bruce G Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3-4):351–379, 1975.

[196] Mervyn Singer, Clifford S. Deutschman, Christopher Warren Seymour, Manu Shankar-Hari, Djillali Annane, Michael Bauer, Rinaldo Bellomo,

Gordon R. Bernard, J P Chiche, Craig M. Coopersmith, Richard S Hotchkiss, Mitchell M. Levy, John C. Marshall, Greg S Martin, Steven M. Opal, Gordon David Rubenfeld, Tom van der Poll, Jean-Louis Vincent, and Derek C. Angus. The third international consensus definitions for sepsis and septic shock (sepsis-3). *JAMA*, 315 8:801–10, 2016.

[197] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 926–934, 2013.

[198] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[199] Michael Spivak. *A Comprehensive Introducion to Differential Geometry*. Publish or Perish, 1970.

[200] Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011.

[201] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles A. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NIPS*, 2017.

[202] Ioan Stanculescu, Christopher K. I. Williams, and Yvonne Freer. A hierarchical switching linear dynamical system applied to the detection of sepsis in neonatal condition monitoring. In *UAI*, 2014.

[203] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

[204] GW Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409, 1980.

[205] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[206] Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286. Springer, 2014.

[207] Harini Suresh, Nathan Hunt, Alistair Edward William Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding using deep networks. *CoRR*, abs/1705.08498, 2017.

[208] Dougal J Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. *ICLR*, 2017.

[209] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[210] Terry Tao. Two small facts about lie groups. https://terrytao.wordpress.com/2011/06/25/two-small-facts-about-lie-groups/, 2011.

[211] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. URL http://arxiv.org/abs/1511.01844.

[212] Paul Thottakkara, Tezcan Ozrazgat-Baslanti, Bradley B Hupf, Parisa Rashidi, Panos M. Pardalos, Petar Momcilovic, and Azra Bihorac. Application of machine learning techniques to high-dimensional clinical data to forecast postoperative complications. In *PloS one*, 2016.

[213] T. Tieleman and G. Hinton. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

[214] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning (ICML)*, pages 1064–1071, 2008.

[215] Jack Ven Tu and M R Guerriere. Use of a neural network as a predictive instrument for length of stay in the intensive care unit following cardiac surgery. *Proceedings. Symposium on Computer Applications in Medical Care*, pages 666–72, 1992.

[216] Oncel Tuzel, Fatih Porikli, and Peter Meer. Learning on lie groups for invariant detection and tracking. In *CVPR 2008*. IEEE, 2008.

[217] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[218] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

[219] Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432, 2017.

[220] J. L. Vincent, Rui Moreno, Jukka Takala, Sheila M. Willatts, Alexandre de Mendonça, Hilgo Bruining, Chantal Reinhart, Peter M. Suter, and Lambertius G. Thijs. The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. *Intensive Care Medicine*, 22:707–710, 1996.

[221] Jean-Louis Vincent. Critical care - where have we been and where are we going? In *Critical care*, 2013.

[222] Jean-Louis Vincent and Daniel De Backer. Circulatory shock. *New England Journal of Medicine*, 369(18):1726–1734, 2013. doi: 10.1056/NEJMra1208943. URL `https://doi.org/10.1056/NEJMra1208943`. PMID: 24171518.

[223] Shyam Visweswaran, Derek C. Angus, Margaret Hsieh, Lisa A. Weissfeld, Donald Yealy, and Gregory F. Cooper. Learning patient-specific predictive models from clinical data. *Journal of biomedical informatics*, 43 5:669–85, 2010.

[224] Kiri Wagstaff. Machine learning that matters. *CoRR*, abs/1206.4656, 2012.

[225] NHS Wales. Together for health–a delivery plan for the crit-

ically ill. 2013. http://www.wales.nhs.uk/documents/delivery-plan-for-the-critically-ill.pdf, 2016.

[226] Jason A. Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association : JAMIA*, 2017.

[227] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3462–3471, 2017.

[228] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of biomedical informatics*, 87:12–20, 2018.

[229] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[230] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics*, pages 1591–1601, 2014.

[231] David Warde-Farley. Improving generative adversarial networks with denoising feature matching. 2017.

[232] Max Harry Weil and Wanchun Tang. From intensive care to critical care medicine: a historical perspective. *American journal of respiratory and critical care medicine*, 183 11:1451–3, 2011.

[233] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1366–1371, 2013.

[234] C. Widmer and G. Rätsch. Multitask Learning in Computational Biology. *JMLR W&CP. ICML 2011 Unsupervised and Transfer Learning Workshop.*, 27: 207–216, 2012.

[235] Mike Wu, Marzyeh Ghassemi, Mengling Feng, Leo A. Celi, Peter Szolovits, and Finale Doshi-Velez. Understanding vasopressor intervention and weaning: risk prediction in a public heterogeneous clinical time series database. *Journal of the American Medical Informatics Association : JAMIA*, 24 3:488–495, 2017.

[236] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM, 2014.

[237] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and

Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

[238] Yanbo Xu, Siddharth Biswal, Shriprasad R. Deshpande, Kevin O. Maher, and Jimeng Sun. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *KDD*, 2018.

[239] Bishan Yang, Wentau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=241703`.

[240] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*, 2017.

[241] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. 18 September 2016.

[242] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Association for Computational Linguistics (ACL)*, pages 545–550, 2014.

[243] Zhiguo Yu, Byron C. Wallace, Todd R. Johnson, and Trevor Cohen. Retrofitting concept vector representations of medical concepts to improve estimates of semantic similarity and relatedness. *Studies in health technology and informatics*, 245:657–661, 2017.

[244] Aaron Zalewski, William J. Long, Alistair Edward William Johnson, Roger G. Mark, and Li wei H. Lehman. Estimating patient's health state

using latent structure inferred from clinical time series and text. *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 449–452, 2017.

[245] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *ICML*, 2017.